Logical Filtering

Report No. UIUCDCS-R-2005-2500

Eyal Amir
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
eyal@cs.uiuc.edu

Stuart Russell
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720-1776, USA
russell@cs.berkeley.edu

Abstract

Filtering denotes any method whereby an agent updates its *belief state*—its knowledge of the state of the world—from a sequence of actions and observations. In *logical filtering*, the belief state is a logical formula describing possible world states and the agent has a (possibly nondeterministic) logical model of its environment and sensors. This paper presents efficient logical filtering algorithms that maintain a compact belief state representation indefinitely, for a broad range of environment classes including nondeterministic, partially observable STRIPS environments and environments in which actions *permute* the state space. Efficient filtering is also possible when the belief state is represented using prime implicates, or when it is approximated by a logically weaker formula.

1 Introduction

Any agent operating in a partially observable environment must perform computations that distinguish among the *a priori* possible current states of the world on the basis of past observations and actions. These computations may operate directly on a representation of the action–observation sequence (e.g., [Winslett, 1990; Kautz *et al.*, 1996]); they may reduce queries about the current state to queries about the initial state (e.g., [Reiter, 2001]); or, they may update the *belief state* (the agent's knowledge about the state of the world) after each action and observation. This latter approach, called *filtering* or *recursive state estimation* in the control theory literature, is particularly useful with unbounded sequences of actions and observations.

The main computational difficulties are 1) the time needed to update the belief state, and 2) the space required to represent it. These depend on the nature of the *transition model*, which describes how the environment evolves over time, the *observation model*, which describes the way in which the environment generates observations, and the family of representations used to denote belief states. Early work, beginning with Gauss, assumed *stochastic* models. For example, the *Kalman filter* [Kalman, 1960] is a ubiquitous device that maintains a multivariate Gaussian belief state over n variables, assuming linear–Gaussian transition and observation model. Crucially, the $O(n^3)$ update cost and the $O(n^2)$ space requirement *do not depend on the length of the observation sequence*; hence, a Kalman filter can run indefinitely. In this paper, we are interested in developing analogous results in the context of logical representations.

We adopt a simple logical language (Section 2) for describing the transition and observation models; the observations and the belief state itself are also logical formulae. The initial state may be only partially known; the transition model, which allows for actions by the agent itself, may be nondeterministic; and the observation model may be nondeterministic and *partial*, in the sense that the agent may not be able to observe the actual state.

Even when we restrict ourselves to propositional logic, it is clear that the general filtering problem is nontrivial (we prove it is computationally hard), because there are exponentially many possible states. We identify several classes of models that allow efficient filtering with respect to the belief-state representation size. Our primary method is based on decomposition theorems showing that 1) filtering distributes over disjunction in the belief state formula, and 2) filtering distributes over conjunction and negation if the actions are *permutations* of the state space. Such actions serve as one-to-one mappings between states, for those states in which they can be applied. We obtain efficient, exact algorithms for DNF belief states and for NNF (Negation Normal Form - all negations are in front of atoms) and CNF belief states with permuting actions. In other cases, we obtain efficient algorithms for *approximate filtering*.

In another class of dynamic systems, we can filter efficiently if the belief state is represented in CNF that includes all its prime implicates. Finally, we show that STRIPS models (possibly with nondeterministic effects of actions) also admit efficient filtering. The STRIPS assumption, that every action has no conditional effects and that an effect's preconditions are the preconditions for the action's execution, is key to this efficiency.

With respect to maintaining a compact representation, we show that properties similar to those mentioned above allow us to filter k-CNF formulae (CNF with clauses of at most k literals, when k is fixed) such that the result is represented in k-CNF (for the same fixed k). Thus, the belief state is maintained in $O(n^k)$ space indefinitely. In particular, we show mild conditions under which a compact belief state can be

maintained in nondeterministic STRIPS domains and in permutation domains. Finally, we show that DNF belief states remain compact if the effects of actions are deterministic and guaranteed to hold. These results are the first analogues, in the logical arena, of the desirable properties possessed by Kalman filters for continuous variables.

Ours is by no means the first work on filtering in a logical context. Early on, it was pointed out that filtering is easy for deterministic systems with a known initial state [Fikes *et al.*, 1972; Lin and Reiter, 1997]. Filtering in nondeterministic domains is more difficult. In particular, the related problem of temporal projection is coNP-hard when the initial state is not fully known, or when actions have nondeterministic effects [Liberatore, 1997] (see also Section 3.3).

Traditionally, computational approaches for filtering take one of three approaches: 1) enumerate the world states possible in every belief state and update each of those states separately, together generating the updated belief state [Ferraris and Giunchiglia, 2000; Cimatti and Roveri, 2000], 2) list the sequence of actions and observations and prove queries on the updated belief state [Reiter, 2001; Sandewall, 1994], or 3) approximate the belief state representation [Son and Baral, 2001; Williams and Nayak, 1996].

The first two approaches cannot be used when there are too many possible worlds (e.g., when the domain includes more than a few dozens of fluents and there are more than 2^{40} possible states) or when the sequence of actions is long (e.g., more than 100 actions). Examples include robot localization, tracking of objects and their relationships, and data mining. The third approach gives rise to many mistakes that are sometimes dangerous, and requires an approximation that fits the given problem (if one exists). Many domains of 100 fluents or less are still computationally infeasible for it.

2 Logical Filtering

In this section we define logical filtering using a transition model and action semantics that are compatible with the *standard semantics* belief update operator of [Winslett, 1990]. (To avoid confusion, this is different from another operator presented in the same publication, *PMA*, that applies a *nonmonotonic* approach to formalize minimal change.) This operator is simple and allows us to examine computational properties easily. It can represent any logical transition system, and specifications in other action languages can be compiled into it [Winslett, 1990; Doherty *et al.*, 1998].

In what follows, for a set of propositional formulae, Ψ, $L(\Psi)$ is the signature of Ψ, i.e., the set of propositional symbols that appear in Ψ. $\mathcal{L}(\Psi)$ is the language of Ψ, i.e., the set of formulae built with $L(\Psi)$. Similarly, $\mathcal{L}(L)$ is the language of L, for a set of symbols L.

A transition system is a tuple $\langle \mathcal{P}, \mathcal{S}, \mathcal{A}, \mathcal{R} \rangle$, where

- $\mathcal{P}$ is a finite set of propositional fluents;
- $\mathcal{S} \subseteq Pow(\mathcal{P})$ is the set of world states;
- $\mathcal{A}$ is a finite set of actions;
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the transition relation.

The intuition for this transition system description is that $\mathcal{P}$ is the set of features that are available for us in the world, every element in $\mathcal{S}$ is a *world state* (i.e., a subset of $\mathcal{P}$, containing propositions that are true in this world state), $\mathcal{A}$ is the set of actions in the system and $\mathcal{R}(s, a, s')$ means that state s' is a possible result of action a in state s.

A *belief state* is a set of world states $\sigma \subseteq \mathcal{S}$. Performing an action a in a belief state σ results in a belief state that includes all the world states that may result from a in a world state in σ. We do not introduce *observations* in this transition model. Instead, we assume that observations are given to us (if at all) as logical sentences after performing an action.

A logical nondeterministic domain description D is a finite set of statements of the following kinds: *value propositions* of the form "**initially** F" describe the initial state and *effect rules* of the form "a **causes** F **if** G" describe the effects of actions, for F and G being *state formulae* (propositional combinations of fluent names). We say that F is the *head* and G is the *tail* of those rules.

For a domain description D we define $\mathcal{P}_D$, $\mathcal{A}_D$ to be the set of propositional fluents and actions mentioned in D, respectively. For a domain description D we define a transition relation $\mathcal{R}_D(s, a, s')$ as follows.

- A fluent $f \in \mathcal{P}_D$ is *possibly affected* by action a in state s, if there is a rule "a **causes** F **if** G" in D such that G is true in s and $f \in L(F)$.
- Let $I(a, s)$ denote the set of fluents in $\mathcal{P}_D$ that are *not* possibly affected by action a in state s.
- Let $F(a, s)$ be a set of all the heads of activated effect rules in s (i.e., if "a **causes** F **if** G" is activated in s, then $F \in F(a, s)$). We consider the case of $F(a, s) = \emptyset$ (no activated effect rules) as $F(a, s) \equiv TRUE$.
- Define (recalling that world states are sets of fluents)

$$\mathcal{R}_D = \left\{ \langle s, a, s' \rangle \;\middle|\; \begin{array}{l} (s' \cap I(a,s)) = (s \cap I(a,s)) \\ \text{and } F(a,s) \text{ is true in } s' \end{array} \right\} \quad (1)$$

When there is no confusion, we write $\mathcal{R}$ for $\mathcal{R}_D$.

If action a has an effect of *FALSE* in s, then it cannot execute.

In partially observable domains, we update our knowledge as a result of executing an action and collecting observations in the resulting state. The following definition of filtering assumes that σ is a set of world states. We use our transition operator $\mathcal{R}$ to define the resulting belief state from each action. An observation o is a formula in our language.

Definition 2.1 (Logical Filtering Semantics). *Let $\sigma \subseteq \mathcal{S}$ be a belief state. The* filtering *of a sequence of actions and observations $\langle a_1, o_1, \ldots, a_t, o_t \rangle$ is defined as follows:*

1. $Filter[\epsilon](\sigma) = \sigma$;
2. $Filter[a](\sigma) = \{s' \mid \langle s, a, s' \rangle \in \mathcal{R},\ s \in \sigma\}$;
3. $Filter[o](\sigma) = \{s \in \sigma \mid o \text{ is true in } s\}$;
4. $Filter[\langle a_i, o_i, \ldots, a_t, o_t \rangle](\sigma) =$
   $Filter[\langle a_{i+1}, o_{i+1}, \ldots, a_t, o_t \rangle]$
   $(Filter[o_i](Filter[a_i](\sigma)))$.

We call Step 2 progression *with a and Step 3* filtering *with o.*

For example, consider a robot that is in charge of cleaning a room. It can execute an action $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$ which has the single effect rule "a **causes** $\mathit{has}(\mathit{broom}) \wedge \neg in(\mathit{broom}, \mathit{closet})$ **if** $in(\mathit{broom}, \mathit{closet})$". Assume that the robot's belief state is $\sigma = Pow(\mathcal{P})$ (i.e., it considers all states possible). Then, $Filter[a](\sigma) = \{s \in$

$Pow(\mathcal{P}) \mid \neg in(\textit{broom}, \textit{closet})$ is true in $s\}$, i.e., after performing the action a we consider all worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ possible. This is because if we are initially in a state in which $in(\textit{broom}, \textit{closet})$, then the resulting state is one in which $\neg in(\textit{broom}, \textit{closet})$, and if we are initially in a state in which $\neg in(\textit{broom}, \textit{closet})$, then we stay in the same state (thus, still satisfying $\neg in(\textit{broom}, \textit{closet})$). Call this resulting belief state σ'. Now, if an observation $o = \textit{has}(\textit{broom})$ is received, then $Filter[o](\sigma')$ is exactly the set of worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ and $\textit{has}(\textit{broom})$.

3 Filtering Logical Formulae

Approaches to filtering actions and observations that at any stage enumerate the states in a belief state do not scale to large domains. An alternative approach is to perform logical progression in a form similar to the one described by [Lin and Reiter, 1997; McIlraith, 1998]. The difference is that now we wish to do so (efficiently) in the context of nondeterministic actions and observations.

In this section we present a straightforward algorithm that filters belief state formulae directly, but does so in worst-case exponential time. This algorithm serves as a starting point for Section 4, where we propose efficient algorithms. We also present distribution properties of filtering over the logical connectives $\wedge, \vee, \neg$, and examine the theoretical limitations of formula filtering. These will guide us in Section 4, and allow us to present classes of systems that are not subject to those limitations and can be tracked in polynomial time and a compact fashion indefinitely.

3.1 Zeroth-Order Filtering Algorithm

In the rest of the paper we assume that a fixed set of fluents persists for action a in all states in which it has an effect. $\textit{Eff}(a)$ is the set of fluents possibly affected by a, and $\overline{\textit{Eff}(a)}$ is $\mathcal{P} \setminus \textit{Eff}(a)$. Some of the following notation assumes an implicit action, when this causes no confusion.

We represent a belief state σ as a logical formula φ such that a state is in σ iff it satisfies φ. The filtering of a belief state formula with an action and an observation is a formula representing the consequences of our effect rules and observation on states that satisfy our initial formula. We specify this filtering process formally using the following notation.

For a set of effect rules $r_1, \ldots, r_l$ for action a, each of the form "a **causes** F_i **if** G_i", write $F_i' = F_{i[\mathcal{P}/\mathcal{P}']}$, for $\mathcal{P}' = \{f_1', \ldots, f_n'\}$ a set of new symbols for fluents, $[\mathcal{P}/\mathcal{P}']$ a shorthand for $[f_1/f_1', ..., f_n/f_n']$, and $[f/f']$ means that we replace all instances of symbol f in the formula by instances of symbol f'. Here, we view $\mathcal{P}$ as the set of fluents in some time t, and $\mathcal{P}'$ as the same fluents in time $t+1$. We add the following set of rules for action a:

$$\begin{aligned}\mathcal{C} = \{&\text{"}a \textbf{ causes } p \textbf{ if } p\text{"}, \text{"}a \textbf{ causes } \neg p \textbf{ if } \neg p\text{"} \mid p \notin \textit{Eff}(a)\} \\ &\cup \{\text{"}a \textbf{ causes } p \textbf{ if } p \wedge \bar{G}\text{"} \mid p \in \textit{Eff}(a)\} \\ &\cup \{\text{"}a \textbf{ causes } \neg p \textbf{ if } \neg p \wedge \bar{G}\text{"} \mid p \in \textit{Eff}(a)\}\end{aligned}$$

where

$$\bar{G} = \neg G_1 \wedge ... \wedge \neg G_l, \tag{2}$$

the assertion that no precondition of a holds. This has a similar effect to adding frame axioms to a set of effect axioms in an action language. We let $r_1, ..., r_m$ be the complete set of rules for a and call the new rules, $\mathcal{C} = \{r_{l+1}, ..., r_m\}$, *completion rules for* a.

We filter a belief-state formula as follows. (We reuse $Filter[\cdot](\cdot)$ for filtering a belief-state formula.) Let $Cn(\Psi)$ be the set of logical consequences of Ψ (i.e., formulae ψ such that $\Psi \models \psi$), and $Cn^{\mathcal{L}}(\Psi)$ be $Cn(\Psi) \cap \mathcal{L}$, the set of logical consequences of Ψ in the language $\mathcal{L}$. We write $Cn^{L}(\Psi)$, when L is a *set of symbols*, to mean $Cn^{\mathcal{L}(L)}(\Psi)$.

1. $Filter[a](\varphi) = (Cn^{\mathcal{P}'}(\varphi \wedge \bigwedge_{i \le m}((\varphi \Rightarrow G_i) \Rightarrow F_i')))_{[\mathcal{P}'/\mathcal{P}]}$;
2. $Filter[o](\varphi) = \varphi \wedge o$.

For example, consider action $a = \textit{fetch}(\textit{broom}, \textit{closet})$ which has the single effect rule "a **causes** $\textit{has}(\textit{broom}) \wedge \neg in(\textit{broom}, \textit{closet})$ **if** $in(\textit{broom}, \textit{closet})$", and consider a belief state formula $\varphi = \textit{TRUE}$ (i.e., we consider all states possible). Then, $Filter[a](\varphi) \equiv \neg in(\textit{broom}, \textit{closet})$, i.e., after performing the action a we consider all worlds that satisfy $\neg in(\textit{broom}, \textit{closet})$ possible (similar to our example in Section 2). Call this resulting belief state formula φ'. Now, if an observation $o = \textit{has}(\textit{broom})$ is received, then $Filter[o](\varphi') \equiv \neg in(\textit{broom}, \textit{closet}) \wedge \textit{has}(\textit{broom})$.

The following generalizes a theorem presented in [Doherty *et al.*, 1998] by allowing conditional and inconsistent effects.

Theorem 3.1. *If φ is a belief state formula, and a an action, then*

$$Filter[a](\{s \in \mathcal{S} \mid s \textit{ satisfies } \varphi\}) = \{s \in \mathcal{S} \mid s \textit{ satisfies } Filter[a](\varphi)\}$$

Proof. See Section A.1 □

Our zeroth-order algorithm computes $Filter[\langle a_1, o_1, ..., a_t, o_t\rangle](\varphi)$ by iterating the application of filtering of a belief-state formula with an action and an observation. It sets $\varphi_0 = \varphi$ and $\varphi_i = Filter[o_i](Filter[a_i](\varphi_{i-1}))$ recursively for $i > 0$ using the two equalities defined above. This algorithm is correct, as shown by Theorem 3.1. It can be implemented using a consequence finder in a restricted language (e.g., [Simon and del Val, 2001]).

From here on, when we say *filtering* we refer to filtering of a belief-state formula, unless otherwise mentioned.

3.2 Distribution Properties and Permutation

We can decompose the filtering of a formula φ along logical connectives once we establish several distribution properties.

Theorem 3.2 (Distribution over Connectives). *Let a be an action, and let φ, ψ be formulae. Then,*

1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $\models Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$
3. $\models Filter[a](\neg\varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](\textit{TRUE})$

Proof. See Section A.2 □

We can say something stronger for actions that act as *permutations* on the states in $\mathcal{S}$ in which they can be executed.

Definition 3.3 (Permuting Actions). *Action a is* permuting, *if for every state s' there is* at most one *s such that $\mathcal{R}(s, a, s')$.*

Domains that include only permuting actions are called *permutation domain*.

For example, consider $a = \textit{fetch}(\textit{broom}, \textit{closet})$ from above, and assume that we add a second effect rule "a **causes** $FALSE$ if $\neg in(\textit{broom}, \textit{closet})$". Thus, a is not executable unless its first rule's precondition holds. Then, the action is a one-to-one mapping between states, when this mapping is defined (it is not defined when a state maps to no resulting state). If this second rule is not added, then the action is not one-to-one because it maps two different states (in the first we already have the broom and in the second the broom is in the closet) to the same state.

In the same spirit, an action *pickUp*(A, B) that picks up block A from the top of block B is one-to-one when it is possible because we can find a single previous state for every resulting state. The same holds for *putDown*(A, C). Other natural examples include *turning a row* in a Rubik's cube, *flipping a light switch*, and *buying a gallon of gas*. In contrast, turning on the light, setting a Rubik's cube to a particular configuration, and filling up the gas tank are *not permutation actions*. Notice that we allow *different* actions to map different states to the same state (e.g., accelerating by 5*MPH* when driving 40*MPH* results in the same state as when decelerating by 5*MPH* when driving 50*MPH*).

Theorem 3.4 (Distribution for Permutation Domains). *Let a be a permuting action, and let φ, ψ be formulae. Then,*

1. $Filter[a](\varphi \vee \psi) \equiv Filter[a](\varphi) \vee Filter[a](\psi)$
2. $Filter[a](\varphi \wedge \psi) \equiv Filter[a](\varphi) \wedge Filter[a](\psi)$
3. $Filter[a](\neg\varphi) \equiv \neg Filter[a](\varphi) \wedge Filter[a](\textit{TRUE})$

Proof Sketch. See Section A.3. □

3.3 Limitations for Compact Representation

It may be argued that filtering may require only polynomial space, if we limit ourselves to initial belief states that are represented compactly and to actions whose effects and preconditions are represented compactly. In Theorem 3.8 we show the contrary. That is, for every general-purpose representation of belief states there is a dynamic system, an initial belief state, and a sequence of actions after which our belief state representation is exponential in the representation of the initial belief state.

P/poly is a *nonstandard* computational complexity class that includes problems that can be answered in polynomial time, if we are given a polynomial-length *hint* that depends only on the length of the input problem. It is considered likely that NP ∩ co-NP $\not\subseteq$ P/poly. Assume so, and let $f(\sigma)$ ($\sigma \subseteq 2^S$) encode belief states using n propositional symbols.

Theorem 3.5 (Craig's Interpolation Theorem [Craig, 1957]). *If $\alpha \vdash \beta$, then there is a formula γ involving only symbols common to both α and β, such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

Theorem 3.6 ([Boppana and Sipser, 1990]). *Assume that for every propositional implication $A \models B$ there is a Craig interpolant C such that $A \models C$ and $C \models B$ and $|C| \leq poly(|A| + |B|)$. Then NP ∩ co-NP ⊆ P/poly.*

Consequently, if NP ∩ co-NP $\not\subseteq$ P/poly, then there is a dynamic system, a belief state σ that can be represented compactly, and a *short* sequence of actions (without observations) such that the result of filtering σ with that sequence has no compact representation. This is stated precisely in the following theorem. In the following we denote the length (in symbol instances) of formula φ by $|\varphi|$, and similarly for a domain description D.

Theorem 3.7. *Let $poly(n)$ be a polynomial function of n. Assume that for every dynamic system D, belief state representation φ_0, and sequence of actions $a_1, ..., a_m$ in D there is a belief state representation φ_i of the belief state after action a_i such that $|\varphi_i| \leq poly(|\varphi_0| \cdot |D| \cdot i)$. Then, NP ∩ co-NP ⊆ P/poly.*

Proof. See appendix A.13. □

This theorem guarantees the existence of at least one system whose belief state representation grows rapidly. How simple can this system be and still hold this bad property? It turns out that there is such a system (in fact, many systems) that is deterministic, and with actions whose effects and preconditions are limited to a few fluents.

Theorem 3.8. *There is dynamic system D with n fluents, belief state σ_0, and action sequence $a_1, ..., a_n$ such that, for all $i \leq n$, $\sigma_i = Filter[a_i](\sigma_{i-1})$, and $|f(\sigma_n)| > poly(|f(\sigma_0)| \cdot |D| \cdot n)$. ($|D|$ is the representation size of D.)*

Proof. See appendix A.14. □

The proof of this theorem reduces the problem of representing the belief state after performing an action to that of representing a *Craig Interpolant*.

4 Efficient and Indefinitely Compact Filtering

In this section we present the main contribution of this paper, namely, a polynomial-time algorithm that computes logical filtering exactly for a significant class of transition systems. For some special cases we present simpler algorithms that are even more efficient. For systems that do not fall within this class our algorithm gives an approximation to the filtering. Also, we show that we can keep the representation of the filtered belief state compact indefinitely for a class of dynamic systems. This class includes nondeterministic STRIPS systems and some systems whose actions are permuting.

The theorems that we consider most important are the following three. Let $|\varphi|$ be the number of literals in φ's representation, let $\#rules(a)$ be the number of effect rules defining a, and let $\bar{G}_a = \bar{G}$ be the precondition of a as defined in (2). Assume that the effect rules that define a given action have identical sets of affected propositional symbols (however, different actions may have different such sets).

Theorem 4.1 (Efficiency). *Given NNF formula φ, action a, and observation o, the filtering algorithm in Figure 1 returns the filtering of φ with a and o, if a is permuting. If a is not permuting, then the algorithm returns an logically weaker formula than the filtering. It does so in time $O(|\varphi| \cdot 2^{\#rules(a)+|L(\bar{G}_a)|})$.*

Theorem 4.2 (Compact Filtering). *Given k-CNF formula φ (fixed k),* deterministic *action a, and observation o in k-CNF,*

the filtering of φ with a and o is in k-CNF, if a is permuting, $Filter[a](TRUE)$ is in k-CNF, and for every literal ℓ in φ, $Filter[a](\ell) \equiv Filter[a](TRUE) \wedge T$, where T is a conjunction of literals.

If a is not permuting, then the algorithm in Figure 1 still returns a formula in k-CNF.

The conditions on action a in the last theorem hold, e.g., for actions whose every defining rule has a precondition that is a single clause (e.g., a literal). It also holds for actions which are defined by at most two rules, and actions that affect all the literals that appear in their preconditions.

Finally, for nondeterministic STRIPS actions (actions that have no conditional effects) we get efficiency and compact representation. Let k-PI-CNF denote the class of k-CNF formulae that include all their prime implicates (i.e., φ in k-PI-CNF iff φ in k-CNF and for every clause C, if $\varphi \models C$, then there is a clause C' in φ such that C' subsumes C). Every formula can be represented in k-PI-CNF for some k.

Theorem 4.3 (STRIPS Actions). *Given k-PI-CNF formula φ (fixed k),* STRIPS *action a with effect in k-PI-CNF, and observation o in 2-CNF, the filtering of φ with a and o is computed in time $O(|\varphi| \cdot k + 2^{|L(\bar{G}_a)|})$. The result is in k-PI-CNF, if after observing o we know if a succeeded or failed.*

4.1 Closed-Form Solution and Factored Filtering

Our zeroth-order filtering algorithm uses consequence finding tools which do not scale to large domains. The following always holds and suggests a different reasoning procedure.

$$Filter[a](\varphi) \equiv \bigwedge_{i_1,\ldots,i_u \leq m,\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}} (F_{i_1} \vee \ldots \vee F_{i_u}). \quad (3)$$

We can compute $Filter[a](\varphi)$ by testing queries of the form $\varphi \models G_{i_1} \vee \ldots \vee G_{i_u}$ (instead of applying consequence finding). On the other hand, it requires an exponential number in m of such tests (recall, m is the number of rules, including the completion rules). Since $m > 2n$ (there are two completion rules for each of the n domain features), this is worse computationally than the method of enumerating all the states.

In what follows we use the intuition that we need only few rules if φ includes only a small subset of $\mathcal{P}$, the fluent symbols of our domain. In general, φ may include many fluent symbols because we may know many things about many different parts of our domain. Nevertheless, if we can decompose φ into small parts that can be filtered separately, then each of the parts includes only a small subset of $\mathcal{P}$, and filtering each of the parts separately becomes easy.

Assume that we order the rules of a such that $r_1, ..., r_t$ ($t \leq l$) satisfy $L(F_i) \cap L(G_i) = \emptyset$ ($r_i =$ "a **causes** F_i **if** G_i", $i \leq t$), and $r_{t+1}, ..., r_l$ satisfy $L(F_i) \cap L(G_i) \neq \emptyset$. Furthermore, let r_{m+1} be the additional rule "a **causes** $\bar{G}$ **if** $\bar{G}$" (recall, m is the number of rules of a, including completion rules). We define $\mathcal{B}$ to be

$$\mathcal{B} = \bigwedge_{i \leq t} (\neg G_i \vee F_i) \wedge \bigwedge_{i_1,\ldots,i_u \in \{t+1,\ldots,l,m+1\}} (\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f \leq u} F_{i_f}) \quad (4)$$

where $\widetilde{G}_{i_1,\ldots,i_u} \equiv Cn^{\overline{Eff(a)}}(\bigwedge_{f \leq u} \neg G_{i_f})$.

$\mathcal{B}$ is a term that is always implied by $Filter[a](TRUE)$, i.e., the progression of zero knowledge with the action a. The first set of conjuncts of $\mathcal{B}$ is the result of applying a rule "a **causes** F_i **if** G_i" whose preconditions are not affected by executing a. Even when we know nothing before performing a, we will know that either the effect occurred or the precondition did not hold and still does not hold. The second set of conjuncts applies a similar intuition for the case of effect rules that may affect the truth value of their original preconditions.

Define $\mathcal{C}(L)$ to be the set of completion rules of a for fluents in L, i.e., $\mathcal{C}(L) = \{i > l \mid$ the effect of $r_i \in \mathcal{C}$ is in $L\}$.

We can now state the fundamental theorem of this Section. It holds for all domains expressed using our action language.

Theorem 4.4 (Closed-Form Representation). *If φ is a belief state formula and $\{r_1, \ldots, r_l\}$ is the set of effect rules for action a, each of the form "a **causes** F_i **if** G_i", then*

$$Filter[a](\varphi) \equiv \bigwedge_{\substack{i_1,\ldots,i_u \in \{1,\ldots l, m+1\} \cup \mathcal{C}(L(\varphi)),\\ \varphi \models G_{i_1} \vee \ldots \vee G_{i_u}}} (F_{i_1} \vee \ldots \vee F_{i_u}) \bigwedge \mathcal{B} \quad (5)$$

Proof. See Section A.4. □

The intuition for equation (5) is that progressing φ with an action a can be computed by looking at all the possible combination of preconditions of effect rules and completion rules for $L(\varphi)$. If we can prove that $G_1 \vee G_2$ holds from φ, then we can conclude that $F_1 \vee F_2$ holds in the result of executing a. The conclusions that are not accounted for with this intuition are the effects that we infer from the completion rules in $\mathcal{C}(L(G_1, ..., G_l))$ together with the effect rules for a. Those conclusions are summarized in $\mathcal{B}$, which is independent of φ.

PROCEDURE NNF-Filter($\langle a_i, o_i \rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, a_i an action, o_i an NNF observation, φ a belief-state formula.
1. If $t = 0$, return φ.
2. Set $\mathcal{B}$ as in equation (4) for a_t but in NNF form.
3. Return $o_t \wedge \mathcal{B} \wedge$ NNF-ProgressStep(a_t, NNF-Filter($\langle a_i, o_i \rangle_{0<i\leq(t-1)}$,$\varphi$)).

PROCEDURE NNF-ProgressStep(a,φ)
a an action. φ a belief-state formula, $r_1, ..., r_l, r_{m+1}$ rules for a.
1. If φ is a literal, then return the NNF form of $\bigwedge\{\bigvee_{i\in I} F_i \mid I \subseteq \{1, ..., l, m+1\} \cup \mathcal{C}(L(\varphi)),\ \varphi \models \bigvee_{i\in I} G_i\}$.
2. If $\varphi = \varphi_1 \vee \varphi_2$, then return NNF-ProgressStep(a, φ_1) $\vee$ NNF-ProgressStep(a, φ_2).
3. It must be that $\varphi = \varphi_1 \wedge \varphi_2$. Return NNF-ProgressStep(a, φ_1) $\wedge$ NNF-ProgressStep(a, φ_2).

Figure 1: Filtering an NNF formula.

Our NNF filtering algorithm is presented in Figure 1. It is much faster than our zeroth-order algorithm, and it relies on Theorems 4.4, 3.2, and 3.4. In the following, if ψ is a logically weaker formula than our filtering, we say that it is a *safe approximation* for filtering. We denote effects by F_i and preconditions by G_i. For an action a, set $t = |\bigcup_{i\leq l} L(G_i)|$.

Corollary 4.5 (NNF Filtering). *Let φ be a formula in NNF with h literals, and let a be an action with l effect rules. Then,*

NNF-Filter safely approximates $Filter[a](\varphi)$ in time $O(h \cdot 2^{l+t})$. If a is permuting, then this computation is exact.

Proof. See Section A.5 □

Extended Example

As an example, consider our room-cleaning robot from above, and the action $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$, with the slightly more elaborate effect rule "a **causes** $\mathit{has}(\mathit{broom}) \wedge \neg in(\mathit{broom}, \mathit{closet})$ **if** $in(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$". Assume that our robot has its belief state represented by $\neg \mathit{locked}(\mathit{closet}) \wedge (in(\mathit{broom}, \mathit{closet}) \vee in(\mathit{broom}, \mathit{shed}))$.

NNF-Filter filters each of the literals separately with a and combines the results. For a given literal, filtering is mostly done is step 1 of NNF-ProgressStep. We describe how this is done for each of the literals in our robot's belief state.

NNF-ProgressStep(a,$\neg \mathit{locked}(\mathit{closet})$) tests in step 1 all possible entailments of the form $\neg \mathit{locked}(\mathit{closet}) \models \bigvee_{i \in I} G_i$, where I includes elements from the effect rule of a, the frame rule for $\bar{G}$ (rule r_{m+1}), and the completion rule for $\neg \mathit{locked}(\mathit{closet})$. For brevity, denote the effect of a by F_1, and the precondition for that rule by G_1. There is only one effect rule defining a, so in this case $\bar{G} = \neg G_1$.

In that step, one of our tests finds that

$$\neg \mathit{locked}(\mathit{closet}) \models G_1 \vee \bar{G}$$

(because $G_1 \vee \bar{G} = G_1 \vee \neg G_1 \equiv TRUE$). This makes us include $F_1 \vee \bar{G}$ in the filtering. In the same step, another test finds that $\neg \mathit{locked}(\mathit{closet}) \models \neg \mathit{locked}(\mathit{closet})$ (trivially true). This makes us include $\neg \mathit{locked}(\mathit{closet})$ in the filtering. The conjunction of the two is logically equivalent to $\neg in(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$. This is the result of filtering $\neg \mathit{locked}(\mathit{closet})$ with a.

Similarly, for $in(\mathit{broom}, \mathit{closet})$, one of our tests in step 1 of NNF-ProgressStep finds that

$$in(\mathit{broom}, \mathit{closet}) \models G_1 \vee (\bar{G} \wedge in(\mathit{broom}, \mathit{closet}))$$

(because $G_1 \vee (\bar{G} \wedge in(\mathit{broom}, \mathit{closet})) \equiv G_1 \vee in(\mathit{broom}, \mathit{closet})$, and the latter follows trivially from $in(\mathit{broom}, \mathit{closet})$). As before, we also find that $in(\mathit{broom}, \mathit{closet}) \models G_1 \vee \bar{G}$. We find that the filtering of $in(\mathit{broom}, \mathit{closet})$ with a is $(in(\mathit{broom}, \mathit{closet}) \vee \mathit{has}(\mathit{broom})) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. Notice that this belief state implies $\mathit{has}(\mathit{broom}) \vee \mathit{locked}(\mathit{closet})$.

For the filtering of the last literal, $in(\mathit{broom}, \mathit{shed})$, we find its filtering to be equivalent to $in(\mathit{broom}, \mathit{shed}) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. (Notice, that we do not know that the broom is not in the closet because initially we did not know that the broom *cannot* be in more than one place; we made this choice to simplify this example.)

Now, we recurse back in the algorithm and combine the filtered formulae. We find that the filtering of $in(\mathit{broom}, \mathit{closet}) \vee in(\mathit{broom}, \mathit{shed})$ is equivalent to $(in(\mathit{broom}, \mathit{shed}) \vee in(\mathit{broom}, \mathit{closet}) \vee \mathit{has}(\mathit{broom})) \wedge (\neg in(\mathit{broom}, \mathit{closet}) \vee \mathit{locked}(\mathit{closet}))$. Notice that this belief state implies $in(\mathit{broom}, \mathit{shed}) \vee \mathit{has}(\mathit{broom}) \vee \mathit{locked}(\mathit{closet})$.

Finally, for the original belief state, $\neg \mathit{locked}(\mathit{closet}) \wedge (in(\mathit{broom}, \mathit{closet}) \vee in(\mathit{broom}, \mathit{shed}))$, we conjoin the last result with the filtering of $\neg \mathit{locked}(\mathit{closet})$ and get

$$(in(\mathit{broom}, \mathit{shed}) \vee \mathit{has}(\mathit{broom})) \wedge \neg in(\mathit{broom}, \mathit{closet}) \wedge \neg \mathit{locked}(\mathit{closet})$$

4.2 DNF and CNF Belief States

If φ is in DNF (a disjunction of conjunctions), then we can limit the size of the resulting formula. We write $D \wedge a \models \psi$ to say that action a has effect rules in D that logically imply ψ in a state in which a is executed (e.g., $\neg \bar{G}$ may be a tautology).

Corollary 4.6 (Iterating DNF Filtering). *Let φ be in DNF with h literals and s disjuncts, and a an action with l effect rules with $\{F_i\}_{i \leq l}$ in DNF with d disjuncts total, and $\{G_i\}_{i \leq l}$ in CNF, each with c conjuncts. Then, $Filter[a](\varphi)$ in DNF is computed exactly in time $O(h \cdot 2^{l+t})$ with at most $\max(2s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts. If $D \wedge a \models \neg \bar{G}$, then it has at most $\max(s, 1) \cdot \binom{d}{d/2} \cdot c^l$ disjuncts.*

Proof. See Section A.6. □

Thus, when a is a *deterministic* action (every rule's effect is a conjunction of literals) with a single effect rule that is always guaranteed to succeed, then the number of disjuncts in the formula does not grow as the filtering progresses.

For CNF formulae we can find a more significant class of actions that allow us to maintain compact representation. We show that under some conditions every k-CNF formula is filtered into a k-CNF formula (fixed k). This implies that the belief state representation is no larger than $(2n)^k$, which is manageable for small fixed k's.

The main observation that we use is that a clause of k literals may give rise to a larger clause after filtering, only if one of the following holds: (a) the filtering of $TRUE$ includes a clause of more than k literals; or (b) the filtering of a literal includes a clause of 2 or more literals that is not subsumed by $Filter[a](TRUE)$. The first case can occur when we do not know whether the action succeeded or not, and which rules applied if it did. In that case, we know that after the action one of the effects holds or no precondition holds (this yields a formula which may include many disjunctions). The second case can occur when the precondition of a rule includes a conjunction of literals. When we filter a single literal we may get a disjunction in the result (of the form *the effect holds, or the rest of the precondition does not*). When we filter a clause, this may cause the filtering to include a larger clause.

The following theorem describes sufficient conditions for filtering a k-CNF formula into k-CNF, thus keeping the representation compact (k is fixed).

Theorem 4.7 (Filtering a k-CNF Clause). *Let C be a k-CNF clause, and action a have l effect rules, all deterministic. Assume that $\mathcal{B}$ is in k-CNF, and that whenever $f \models \bigvee_{i \in I} G_i$ for literal f and $I \subset \{1, ..., l, m+1\}$, $|I| \geq 2$, then $D \wedge a \models \bigvee_{i \in I} G_i$ or $f \models G_i$ for some $i \in I$. Then, $Filter[a](C)$ is in k-CNF and can be computed in time $O(2^{l+t})$.*

Proof. See Section A.7. □

Note that $\mathcal{B}$ is in k-CNF for an action a when $|\bigcup_{i\le l} L(G_i) \cup \mathit{Eff}(a)| \le k$ or $|\bigcup_{i\le l} L(G_i) \setminus \mathit{Eff}(a)| + l \le k$ or $l = 1$ and $|L(G_1)| - min(1, |L(F_1) \setminus L(G_1)|) \le k$.

Consequently, filtering with actions that are permuting maintains a k-CNF if $\models \bigvee_{i\in I} G_i$ whenever $f \models \bigvee_{i\in I} G_i$ for some $|I| \ge 2$. An example of such an action is *flipping a switch* (if the light is on, it will get turned off, and vice versa). One of the preconditions always holds. Another example is moving a block (whichever it is) from the top of a stack. For literal f, if $f \models \mathit{top}(A) \vee \mathit{top}(B)$, then the disjunction is implied by $D \wedge a$ or a single precondition follows from f.

Corollary 4.8 (Iterating CNF Filtering). *If φ is in k-CNF, then the assumptions of Theorem 4.7 imply that $Filter[a](\varphi)$ is approximated safely in time $O(|\varphi| \cdot 2^{l+t})$, with a result in k-CNF. If a is permuting, then this computation is exact.*

4.3 Prime-Implicate Belief States

It turns out that a form of distribution over conjunction holds for all actions if the belief state is represented as the conjunction of all of its *prime implicates* (formulae we call *prime implicate belief states* (PI-CNF)). In this form we can distribute the computation to the conjuncts and conjoin the result of filtering small subgroups of them separately. More precisely,

$$Filter[a](\varphi) \equiv \bigwedge_{j_1,\ldots,j_z \le s} Filter[a](\bigwedge_{g\le z} C_{j_g}) \qquad (6)$$

for z a number that depends on the representation of the preconditions of a and on the number of rules defining a.

PROCEDURE PI-Filter($\langle a_i, o_i\rangle_{0<i\le t}$,$\varphi$)
$\forall i$, a_i an action, o_i an observation, φ a belief-state formula.
1. If $t = 0$, return φ.
2. Return the PI-CNF form of $o_t \wedge$ PI-ProgressStep(a_t, PI-Filter($\langle a_i, o_i\rangle_{0<i\le(t-1)}$,$\varphi$)).

PROCEDURE PI-ProgressStep(a,φ)
a an action, φ a belief-state formula. $r_1, ..., r_l, r_{m+1}$ rules for a.
1. Let $res \leftarrow$ *TRUE*.
2. For every $i_1, ..., i_j \in \{1, ..., l, m+1\}$, and $f_1, ..., f_u$ literals in $Pre(a) \setminus \mathit{Eff}(a)$, do
   (a) Let $\bigwedge_{g\le z} \hat{C}_g$ be the CNF representation of $\bigvee_{h\le j} G_{i_h}$.
   (b) For every $g \le z$, nondeterministically choose a clause C_g in φ whose restriction to $Pre(a)$ subsumes $\hat{C}_g$.
   (c) If there is a clause for every $g \le z$, then set $res \leftarrow res \wedge F$, where F is $\bigvee_{h\le j}(F_{i_h} \vee (C_{i_h} \cap \overline{\mathit{Eff}(a)}))$.
3. Return res.

Figure 2: Filtering a Prime-Implicate CNF formula.

Theorem 4.9 (Filtering Prime Implicates: DNF Precond.). *Let φ be in k-PI-CNF. Let action a have l effect rules with effects in k-CNF and preconditions in t-DNF with at most d disjuncts. Then, equation (6) holds for $z = t^{l\cdot d}\cdot(l\cdot d+1)$, and PI-Filter (Figure 2) computes $Filter[a](\varphi)$ exactly in time $O(2^{l\cdot|Pre(a)\cup\mathit{Eff}(a)|}\cdot(s^z+z))$. If $D\wedge a \models \neg\bar{G}$, then $z = t^{l\cdot d}$, and the computation takes $O(2^l\cdot(s^z+z))$ time.*

Proof. See Section A.8 □

Theorem 4.10 (Filtering Prime Implicates). *Let φ be in k-PI-CNF (PI-CNF and k-CNF), and let action a have l effect rules with effects in k-CNF and preconditions in d-CNF, with each G_i of at most c clauses. Then, equation (6) holds for $z = c^l\cdot(d^c+1)$, and PI-Filter (Figure 2) computes $Filter[a](\varphi)$ exactly in time $O(2^{l\cdot|Pre(a)\cup\mathit{Eff}(a)|}\cdot(s^z+z))$. If $D\wedge a \models \neg\bar{G}$, then $z = c^l$, with $O(2^l\cdot(s^z+z))$ time.*

Corollary 4.11 (Maintaining k-PI-CNF). *Let φ be a k-PI-CNF formula, and let action a have l effect rules, all deterministic. Assume that G_i is a disjunction of literals, for all $i \le l$. Also, assume that $D\wedge a \models \neg\bar{G}$. Then, $Filter[a](\varphi) \equiv$ PI-ProgressStep(a,φ), and the latter is in k-CNF. If $k = 2$, then PI-Filter(a,φ) is in k-PI-CNF.*

Proof. See Section A.9. □

The conclusion for $k = 2$ uses the fact that every prime implicate of a formula in 2-CNF is a clause with at most two literals. Simple counter examples (omitted) show that k-PI-CNF cannot be maintained without these conditions.

4.4 Nondeterministic STRIPS Domains

STRIPS domains present a special case of the results that we discussed above. In such domains every action has a single rule (no conditional effects) and actions can be executed only when their preconditions hold[1]. Unlike the original STRIPS, we allow nondeterministic effects, and allow belief states to be any CNF formulae in the fluents of the domain.

More precisely, every action a has exactly two effect rules, r_1, r_2. Their preconditions are such that $G_1 \equiv \neg G_2$. Also, $F_2 \equiv FALSE$. Thus, a can be executed only when G_1 holds. Consequently, when we filter with a we assert implicitly that its preconditions held in the last world state.

The assumption that there is only one rule that determines a's effects and otherwise the action is not executed has a dramatic effect. For $l_1, ..., l_k$ literals we get that

$$Filter[a](l_1 \vee ... \vee l_k) \equiv \begin{cases} T_a & \exists i \le k\ l_i \in \mathcal{L}(\mathit{Eff}(a)) \\ T_a \wedge \bigvee_{i\le k} l_i & l_1, ..., l_k \notin \mathcal{L}(\mathit{Eff}(a)) \end{cases} \qquad (7)$$

for $T_a = Filter[a](\mathit{TRUE})$.

Theorem 4.12 (Iterating STRIPS Filtering: CNF). *Let φ be in k-CNF with s clauses and a a STRIPS action. If F_1 is in k-CNF and $|L(G_1) \setminus L(F_1)| \le t$, for some $t \le k$, then $Filter[a](\varphi)$ can be approximated safely in time $O(s\cdot k + 2^t)$, yielding a k-CNF formula. If a is permuting, then this computation is exact.*

Proof. See section A.10. □

If our representation is PI-CNF, then we get an even stronger result, leading to the algorithm in Figure 3.

[1]With the alternate assumption that actions have no effect unless their preconditions hold, but observations (or their absence) are guaranteed to distinguish actions' success from failure, the same results hold, except that now the compactness of representation remains only after filtering with observations.

Theorem 4.13 (Factoring STRIPS filtering: PI-CNF). *Let $\bigwedge_{i\leq s} C_i$ be in PI-CNF, and let a be a STRIPS action. Then,*

$$Filter[a](\bigwedge_{i\leq s} C_i) \equiv \bigwedge_{i\leq s} Filter[a](C_i).$$

Proof. See Section A.11. □

For example, for $a = \mathit{fetch}(\mathit{broom}, \mathit{closet})$ that has effect $F_1 = \mathit{has}(\mathit{broom}) \wedge \neg in(\mathit{broom}, \mathit{closet})$, if we know $(in(broom, closet) \vee \neg locked(closet)) \wedge (in(broom, shed) \vee locked(closet))$ before applying a, then after it we know $F_1 \wedge (in(broom, shed) \vee locked(closet))$.

Corollary 4.14 (Iterating STRIPS filtering: k-PI-CNF). *Let φ be in k-PI-CNF, and let a be a STRIPS action with F_1 in k-PI-CNF, $t = |L(G_1) \setminus L(F_1)|$, and $t \leq k$. Then, STRIPS-Filter(a,φ) computes $Filter[a](\varphi)$ exactly in time $O(|\varphi| \cdot k + 2^t)$, yielding a k-PI-CNF formula.*

Proof. See Section A.12. □

This means that we can filter in practice any prime implicate belief state in any nondeterministic STRIPS domain. This filtering stays compact, with the size depending only on the PI-CNF representation of F_1 and the number of propositional symbols in G_1 but not F_1.

Finally, every STRIPS action can be filtered efficiently, even if we drop the assumption that either the action succeeds or we observe an error. This can be done by assuming that the action succeeds, finding the filtering of that action, and then disjoining the result with the initial belief state. Nonetheless, this scheme may cause representation space explosion as filtering proceeds over multiple steps, unless some care is taken.

PROCEDURE STRIPS-Filter($\langle a_i, o_i \rangle_{0<i\leq t}$,$\varphi$)
$\forall i$, a_i an action, o_i an observation, φ a belief-state formula.
1. For i from 1 to t do,
   (a) Set $\varphi' \leftarrow \bigwedge_{C\in\varphi} Filter[a_i](C)$, where $Filter[a_i](C)$ is computed using (7), and eliminate subsumed clauses.
   (b) Let φ be the prime implicates of $\varphi' \wedge o_i$.
2. Return φ.

Figure 3: Filtering a PI-CNF formula with STRIPS actions.

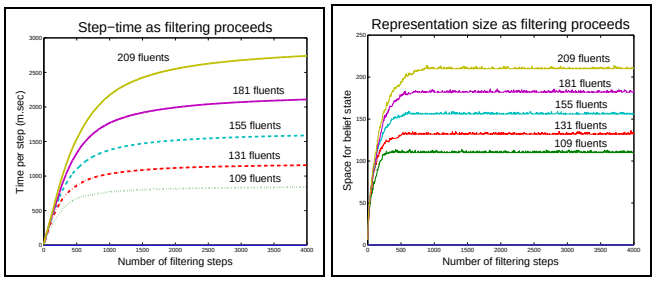

Figure 4: STRIPS-Filter in the blocks world.

We tested our STRIPS-filter algorithm in partially observable blocks-world domains. The implementation in LISP includes a random action-sequence and observation-sequence generator, and both the generator and filtering algorithm receive a description of the domain, actions and observations specified in PDDL (a plan-problem specification language).

We ran the algorithm with blocks-world domains of increasing sizes (3 to 20 blocks), yielding domains that range from tens to over a thousand of propositional features. We collected the time taken per filtering step for each of the executions and the space taken overall at every iteration, starting with zero knowledge. The results are shown in Figure 4.

4.5 Observation Model

An observation model is a theory $\mathcal{O}$ that includes *observation constraints*, axioms that describe the relationship between observed facts and other fluents. We allow $\mathcal{O}$ to include any axioms. Our model assumes that observations o are collected after an action is executed. o is a sentence made up with fluents, similar to the observation constraints. The conjunction, $\mathcal{O} \wedge o$ is then used to filter the resulting belief state. Formally,

$$Filter[o](\varphi) = \varphi \wedge o \wedge \mathcal{O}. \quad (8)$$

This allows all the results that we enumerated above to apply with this model as well. The following connects the results of Sections 3,4 to filtering with observations.

Corollary 4.15. *If $o \wedge \mathcal{O}$ is in k-CNF and $Filter[a](\varphi)$ is in k-CNF, then $Filter[a, o](\varphi)$ is in k-CNF.*

In particular, this means that our results for STRIPS domains and for NNF, CNF and DNF formulae still hold here. Thus, the representation remains compact in the cases that we indicated already, and computation remains easy in the same cases as well. Finally, for k-PI-CNF, we get the following.

Corollary 4.16 (Obs. and k-PI-CNF). *If $o \wedge \mathcal{O}$ is in 2-CNF, and φ is in k-PI-CNF, then $Filter[o](\varphi)$ is in k-PI-CNF.*

5 Conclusions

In this paper we presented the task of logical filtering and gave it a computational treatment. The results we obtained here have implications for monitoring and controlling dynamic systems. In many cases we present a closed-form computation of the filtering and in others show how to approximate this computation. In some cases we can guarantee that the size of the representation of the filtered formula can be bounded and kept small. In those cases, logical filtering can be used to control processes that run over very long periods of time. Examples of such systems are abundant and include robot motion control, natural language processing, and agents that explore partially observed worlds.

We made use of several assumptions in this paper in different contexts and with different consequences. We presented permutation domains and the certainty of existence of an effect ($D \wedge a \models \neg\bar{G}$) as characteristics of the domain that make filtering easier. We showed that the commonly used assumption that every action has a relatively small number of rules (at most polynomial in n), and that effects, preconditions and terms in the belief state typically use a small vocabulary, all have a drastic effect on the computational effort needed for filtering and on the size of the resulting belief state.

The need to track the state of the world is a basic one, and many works have appealed to it implicitly in the past. However, the computational treatment of such tracking has been avoided so far, partially due to the absence of a developed theory of nondeterministic domains, and partially due to negative results about the general cases of this task. Nonetheless, this problem and methods for its solution have received much attention in control theory. The results we obtained here promise to find their application in this domain and may be combined with stochastic filtering techniques.

Acknowledgments

This research was supported by ONR MURI Fund N00014-01-1-0890, ONR MURI Fund N00014-00-1-0637, and NSF grant ECS-9873474. The first author thanks Xuanlong Nguyen for a stimulating discussion on Theorem 3.4.

A Proofs

A.1 Proof of Theorem 3.1

Proof. We show that the two sets of world states have the same elements. We show first that the left-hand side of the equality is contained in the right-hand side.

Take $s' \in Filter[a](\{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\})$. We show that s' satisfies $Filter[a](\varphi)$. From Definition 2.1 there is $s \in \mathcal{S}$ such that $s \in \{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\}$ such that $\langle s, a, s' \rangle \in \mathcal{R}$. In other words, there is $s \in \mathcal{S}$ such that s satisfies φ and $\langle s, a, s' \rangle \in \mathcal{R}$.

To prove that s' satisfies $Filter[a](\varphi)$ we need to show that $\varphi \wedge \bigwedge_{i \leq m}(\varphi \Rightarrow G_i) \Rightarrow F_i'))$. together with the truth assignment s' to $\mathcal{P}'$ is satisfiable. We show that the truth assignment s to $\mathcal{P}$ satisfies this formula together with the truth assignment s' to $\mathcal{P}'$. It is not satisfying this formula only if one of the conjuncts $(\varphi \Rightarrow G_i) \Rightarrow F_i'$ or φ is falsified. This cannot be the case for φ by our choice of s.

Assume by contradiction that this is the case for some i. Then, the truth assignments of s, s' to $\mathcal{P}, \mathcal{P}'$ sanction that $\varphi \Rightarrow G_i$ holds but F_i' does not. From the way we defined $\mathcal{R}$ (i.e., $\mathcal{R}_D$ in equation (1)) we can conclude that $F(a, s)$ is true in s' and that $s' \cap I(a, s) = s \cap I(a, s)$. However, $F(a, s)$ is the conjunction of heads of activated rules and $I(a, s)$ is the set of unaffected fluents. If $i \leq l$ (i.e., r_i is an original rule), then $\varphi \wedge (\varphi \Rightarrow G_i)$ implies that G_i holds in s and the rule r_i is activated. Thus, $F(a, s)$ includes F_i, and F_i is true in s'. This contradicts our assumption that F_i' does not hold with the truth assignment s' to $\mathcal{P}'$. Thus, there is no such conjunct in $\bigwedge_{i \leq m}(\varphi \Rightarrow G_i) \Rightarrow F_i'))$ and the truth assignment s, s' to $\mathcal{P}, \mathcal{P}'$, respectively, satisfies this formula. From the definition of $Filter[a](\varphi)$ and Craig's interpolation theorem for propositional logic (See Theorem A.1) we get that s' satisfies $Filter[a](\varphi)$.

For the opposite direction (showing the right-hand side is contained in the left-hand side), take $s' \in \mathcal{S}$ that satisfies $Filter[a](\varphi)$. We show that $s' \in Filter[a](\{s \in \mathcal{S} \mid s \text{ satisfies } \varphi\})$. From Craig's interpolation theorem for propositional logic we get that there is a truth assignment s for $\mathcal{P}$ such that the truth assignment s, s' to $\mathcal{P}, \mathcal{P}'$, respectively, together satisfy $\varphi \wedge \bigwedge_{i \leq m}(\varphi \Rightarrow G_i) \Rightarrow F_i'))$ (otherwise, there is no such truth assignment, and $Filter[a](\varphi)$ is not satisfiable; in particular, s' does not satisfy it). In a manner similar to the first part of this proof (observing the way $\mathcal{R}$ is defined) we can show that $\mathcal{R}(s, a, s')$ and the second part is done. □

A.2 Proof of Theorem 3.2

Proof. We show this theorem for the set-of-states representation of belief states, and it will follow for the formula-based representation.

1. Take a state s' that satisfies $Filter[a](\varphi \vee \psi)$. Then, there is a state s that satisfies $\varphi \vee \psi$ such that $R_D(s, a, s')$. Thus, s satisfies one of φ or ψ because s is a complete setting of the fluents. Thus, s' is in one of $Filter[a](\varphi)$, $Filter[a](\psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \vee \psi) \Rightarrow Filter[a](\varphi) \vee Filter[a](\psi)$.

For the other direction, take s' that satisfies $Filter[a](\varphi) \vee Filter[a](\psi)$. Then, it satisfies one of $Filter[a](\varphi)$, $Filter[a](\psi)$. Thus, there is a state s such that $R_D(s, a, s')$ and s satisfies one of φ,ψ. Thus, s satisfies $\varphi \vee \psi$ and s' satisfies $Filter[a](\varphi \vee \psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \vee \psi) \Leftarrow Filter[a](\varphi) \vee Filter[a](\psi)$.

2. Take a state s' that satisfies $Filter[a](\varphi \wedge \psi)$. Then, there is a state s that satisfies $\varphi \wedge \psi$ such that $R_D(s, a, s')$. Thus, s satisfies both of φ and ψ. Thus, s' is in both of $Filter[a](\varphi)$, $Filter[a](\psi)$. We conclude that every s' that satisfies $Filter[a](\varphi \wedge \psi)$ also satisfies $Filter[a](\varphi) \wedge Filter[a](\psi)$. From Theorem 3.1 it follows that $Filter[a](\varphi \wedge \psi) \Rightarrow Filter[a](\varphi) \wedge Filter[a](\psi)$.

3. Take s' that satisfies $\neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. Then, there is no state s such that $R_D(s, a, s')$ and s satisfies φ. Thus, for every state s such that $R_D(s, a, s')$ s satisfies $\neg\varphi$. Since s' satisfies $Filter[a](TRUE)$ there is a state s such that $R_D(s, a, s')$. Thus, this s satisfies $\neg\varphi$ and s' satisfies $Filter[a](\neg\varphi)$. From Theorem 3.1 it follows that $Filter[a](\neg\varphi) \Leftarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. □

A.3 Proof of Theorem 3.4

Proof. Theorem 3.2 supplies the proof of 1, the "⇒" direction of 2, and the "⇐" direction of 3. Thus, we are left to prove the "⇐" direction of 2 and the "⇒" direction of 3.

For "⇐" of 2, let s' be a world state that satisfies $Filter[a](\varphi) \wedge Filter[a](\psi)$. Then, it satisfies both of $Filter[a](\varphi)$, $Filter[a](\psi)$. For $Filter[a](\varphi)$ there is a state s such that $R_D(s, a, s')$ and s satisfies φ. Similarly, for $Filter[a](\psi)$ there is a state s_1 such that $R_D(s_1, a, s')$ and s_1 satisfies ψ. However, since a acts as a one-to-one mapping from $\mathcal{S}$ to $\mathcal{S}$, there is only one state in $\mathcal{S}$ that maps to s'. Thus, $s = s_1$, and s satisfies ψ. Thus, s satisfies $\varphi \wedge \psi$ and s' satisfies $Filter[a](\varphi \wedge \psi)$. From Theorem 3.1 it follows that $\models Filter[a](\varphi \wedge \psi) \Leftarrow Filter[a](\varphi) \wedge Filter[a](\psi)$.

For "⇒" of 3, let s' be a world state that satisfies $Filter[a](\neg\varphi)$. Then, there is a state s that satisfies $\neg\varphi$ such that $R_D(s, a, s')$. Thus, s does not satisfies φ. Since a acts as a one-to-one mapping from $\mathcal{S}$ to $\mathcal{S}$, there is only one state that maps to s' after a. Thus, there is no state s_1 that satisfies φ and for which $R_D(s_1, a, s')$. Thus s' does not satisfy $Filter[a](\varphi)$ meaning that it satisfies $\neg Filter[a](\varphi)$.

Clearly, s' also satisfies $Filter[a](TRUE)$. We get that s' satisfies $\neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. From Theorem 3.1 it follows that $\models Filter[a](\neg\varphi) \Rightarrow \neg Filter[a](\varphi) \wedge Filter[a](TRUE)$. □

A.4 Proof of Theorem 4.4

Proof. Let Ψ be the formula on the right-hand side of (5).

Showing that $Filter[a](\varphi) \models \Psi$: Take $F_{i_1} \vee ... \vee F_{i_u}$ that is implied by formula (5). Then, $\varphi \models G_{i_1} \vee ... \vee G_{i_u}$, by the same formula. When we look at the definition of filtering of a formula (Theorem 3.1) we notice that every G_{i_f}, $f \leq u$, appears in this definition or $G_{i_f} = \bar{G}$. The latter belongs to r_{m+1}, which is the only rule that we use in (5) that is not a completion rule or an original rule. However, r_{m+1} follows from the completion rules in $\mathcal{C}$. Thus, $F_{i_1} \vee ... \vee F_{i_u}$ follows from the definition of filtering.

We show that $\mathcal{B}$ follows from $Filter[a](\varphi)$. For a rule r_i with $i \leq t$, we know that $L(G_i) \cap \mathit{Eff}(a) = \emptyset$. Thus, the completion rules of a ensure that $\neg G_i$ holds after executing a if it holds before executing a. However, $\neg G_i \vee G_i$ is a tautology (thus, in particular, it follows from φ), so we get that $\neg G_i \vee F_i$ holds in the result of executing a.

Similarly, for a disjunction of rule preconditions, $\bigvee_{f \leq u} G_{i_f}$ the disjunction $\neg(\bigvee_{f \leq u} G_{i_f}) \vee \bigvee_{f \leq u} G_{i_f}$ is a tautology. From case analysis and Theorem 3.1 we get that in the consequence of executing a we know $\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f \leq u} F_{i_f}$.

Showing that $\Psi \models Filter[a](\varphi)$: We use the equivalence stated in formula (3) by showing that every disjunction $\bigvee f \leq u F_{i_f}$ present in (3) is implied by Ψ.

Take $\bigvee_{f \leq u} F_{i_f}$ to be a conjunct in (3), and assume that it is minimal (i.e., no other conjunct includes a strict subset of F_{i_f}'s). Then, $\varphi \models \bigvee_{f \leq u} G_{i_f}$. If all r_{i_f}, $f \leq u$, are original rules of a (not completion rules) or completion rules for literals in $L(\varphi)$, then they all appear in (5), and therefore $\bigvee_{f \leq u} F_{i_f}$ appears in (5).

W.l.o.g. assume that we ordered i_f such that

- r_{i_f}, $f \leq t$, are original rules for a,
- r_{i_f}, $t < f \leq v$, are completion rules literals in $L(\varphi)$,
- r_{i_f}, $v < f \leq w$, are completion rules for literals in $\mathit{Eff}(a) \setminus L(\varphi)$, and
- r_{i_f}, $w < f \leq u$, are completion rules for literals in $\overline{\mathit{Eff}(a)} \setminus L(\varphi)$.

Denote the literals that are the heads of the completion rules l_{i_f}, $t < f \leq u$, respectively. If $v = u$ (there are no literals of the second and third sort), then we are done, by the previous paragraphs. Thus, assume that $v < u$.

We show that $\bigvee_{f \leq u} F_{i_f}$ is implied by Ψ.

$$\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \bigvee_{v < f \leq w} (l_{i_f} \wedge \bar{G}) \vee \bigvee_{w < f \leq u} l_{i_f}$$

by the way we sorted F_{i_f}, and the fact that the disjunction $\bigvee_{f \leq u} F_{i_f}$ is one of the conjuncts in (3).

Let $\psi = \bigvee_{f \leq v} G_{i_f}$. Then, $\varphi \models \psi \vee (\bar{G} \wedge \bigvee_{v < f \leq w} l_{i_f}) \vee \bigvee_{w < f \leq u} l_{i_f}$. From this we get $\varphi \models \psi \vee \bigvee_{v < f \leq u} l_{i_f}$.

We make use of Craig's interpolation Theorem:

Theorem A.1 ([Craig, 1957]). *Let α, β be sentences such that $\alpha \vdash \beta$. Then there is a formula γ involving only nonlogical symbols common to both α and β, such that $\alpha \vdash \gamma$ and $\gamma \vdash \beta$.*

We know that $L(\bigvee_{v < f \leq u} l_{i_f}) \cap L(\varphi) = \emptyset$. Craig's interpolation theorem implies that $\varphi \models \psi$ or $\psi \vee \bigvee_{v < f \leq u} l_{i_f}$ is a tautology. We assumed minimality of $\bigvee_{f \leq u} F_{i_f}$ in (3), so the first case cannot be.

Thus, it must be that $\psi \vee \bigvee_{v < f \leq u} l_{i_f}$ is a tautology. Placing the meaning of ψ in this formula we get that $\bigvee_{f \leq t} G_{i_f} \vee \bigvee_{t < f \leq u} l_{i_f}$ is a tautology. This means that $\neg(\bigvee_{f \leq t} G_{i_f}) \models \bigvee_{t < f \leq u} l_{i_f}$.

We know that $\neg(\bigvee_{f \leq t} G_{i_f}) \vee \bigvee_{f \leq t} G_{i_f}$ is a tautology because $a \vee \neg a$ is a tautology for every sentence a. We look at two cases:

Case 1: $L(\bigvee_{f \leq t} G_{i_f}) \cap \mathit{Eff}(a) = \emptyset$: From this assumption, for every $f \leq t$ we have $\neg G_{i_f} \vee F_{i_f}$ as a conjunct in $\mathcal{B}$. For each $j \leq t$ we get the implied sentence $\neg G_{i_j} \vee \bigvee_{f \leq t} F_{i_f}$. The conjunction of those sentences for $j \leq t$ implies $\bigwedge_{j \leq t} (\neg G_{i_j} \vee \bigvee_{f \leq t} F_{i_f})$ which is equivalent to $\neg(\bigvee_{f \leq t} G_{i_f}) \vee \bigvee_{f \leq t} F_{i_f}$.

We already concluded above that $\neg(\bigvee_{f \leq t} G_{i_f}) \models \bigvee_{t < f \leq u} l_{i_f}$, so we get that $\bigvee_{t < f \leq u} l_{i_f} \vee \bigvee_{f \leq t} F_{i_f}$ is logically entailed by $\mathcal{B}$ and we are done (this last formula is exactly $\bigvee_{f \leq u} F_{i_f}$ with some replacement of positions of disjuncts).

Case 2: $L(\bigvee_{f \leq t} G_{i_f}) \cap \mathit{Eff}(a) \neq \emptyset$: Our earlier conclusion $\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \bigvee_{v < f \leq w} (l_{i_f} \wedge \bar{G}) \vee \bigvee_{w < f \leq u} l_{i_f}$ implies that $\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \bar{G} \vee \bigvee_{w < f \leq u} l_{i_f}$ or $\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \bigvee_{w < f \leq u} l_{i_f}$, depending on whether $v < w$ or $v = w$, respectively. For the rest of this proof we write θ for $\bar{G}$ or *FALSE*, according to whether $\bar{G}$ appears in this disjunction or not, respectively.

Craig's interpolation theorem implies that either $\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \theta$ or $\bigvee_{f \leq v} G_{i_f} \vee \theta \vee \bigvee_{w < f \leq u} l_{i_f}$ is a tautology. This is because l_{i_f} is not in $L(\varphi)$. We look at these two cases separately.

We assumed minimality of $\bigvee_{f \leq u} F_{i_f}$ in (3), so the first case can be only if $w = u$ (i.e., no literals of the third kind). Thus, in this case, $\varphi \models \bigvee_{f \leq v} G_{i_f} \vee \theta$, which is a precondition for a conjunction in (5). Thus, (5) includes the sentence $\bigvee_{f \leq v} F_{i_f} \vee \theta$ in its conjunction.

Now, we know that $\bar{G} \models \neg(\bigvee_{f \leq t} G_{i_f})$ because $\{G_{i_f}\}_{f \leq t} \subseteq \{G_i\}_{i \leq l}$. Also, we already concluded that $\neg(\bigvee_{f \leq t} G_{i_f}) \models \bigvee_{t < f \leq u} l_{i_f}$. We get that $\bigvee_{f \leq v} F_{i_f} \vee \bar{G}$ logically entails $\bigvee_{f \leq v} F_{i_f} \vee \bigvee_{t < f \leq u} l_{i_f}$ which is equivalent to $\bigvee_{f \leq u} F_{i_f}$. Thus, $\bigvee_{f \leq v} F_{i_f} \vee \theta$ implies $\bigvee_{f \leq u} F_{i_f}$, so (5) implies $\bigvee_{f \leq u} F_{i_f}$ and we are done with this case.

Finally, the second case is that of $\bigvee_{f\leq v} G_{i_f} \vee \theta \vee \bigvee_{w<f\leq u} l_{i_f}$ is a tautology. (Notice that it is possible that we can conclude a stronger disjunction when $v = w$, but we ignore this case because a treatment of the weaker case implies a treatment for this one.)

W.l.o.g. assume that the literals l_{i_f} for $t < f \leq v$ are ordered such that there is $v' \leq v$ with $l_{i_f} \in \mathcal{L}(\mathit{Eff}(a))$ for all f such that $t < f \leq v'$ and $l_{i_f} \in \mathcal{L}(\overline{\mathit{Eff}(a)})$ for all f such that $v' < f \leq v$. Then, the formula $\bigvee_{f\leq v} G_{i_f} \vee \theta \vee \bigvee_{w<f\leq u} l_{i_f}$ is equal to $\bigvee_{f\leq t} G_{i_f} \vee \bigvee_{t<f\leq v'} (l_{i_f} \wedge \bar{G}) \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \theta \vee \bigvee_{w<f\leq u} l_{i_f}$.

Again, we take θ' to be either $\bar{G}$ or *FALSE*, if $\bar{G}$ appears in this formula or not, respectively. This implies that $\bigvee_{f\leq t} G_{i_f} \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \theta' \vee \bigvee_{w<f\leq u} l_{i_f}$ is a tautology.

Consequently, $\neg(\bigvee_{f\leq t} G_{i_f}) \wedge \neg\theta' \models \bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f}$. W.l.o.g., assume that we ordered $\{G_{i_f}\}_{f\leq t}$ such that there is $t' \leq t$ such that $L(\{G_{i_f}\}_{f\leq t'}) \cap \mathit{Eff}(a) = \emptyset$ and $\forall f\ (t' < f \leq t \Rightarrow L(G_{i_f}) \cap \mathit{Eff}(a) \neq \emptyset$. Then, we rewrite the last formula into $\neg(\bigvee_{t'<f\leq t} G_{i_f}) \wedge \neg\theta' \models \bigvee_{f\leq t'} G_{i_f} \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f}$.

Craig's interpolation theorem implies that there is a formula ξ such that $\xi \in \mathcal{L}(\overline{\mathit{Eff}(a)}) \cap \mathcal{L}(\bar{G})$ and $\neg(\bigvee_{t'<f\leq t} G_{i_f}) \wedge \neg\theta' \models \xi$ and $\xi \models \bigvee_{f\leq t'} G_{i_f} \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f}$.

Let $i_{m+1} = m + 1$ if $\theta' = \bar{G}$ or $i_{m+1} = \emptyset$ otherwise. From the definition of $\widetilde{G}_{i_{t'+1},\ldots,i_t,i_{m+1}}$ we get that $\widetilde{G}_{i_{t'+1},\ldots,i_t,i_{m+1}} \models \xi$.

Thus, the formula $\widetilde{G}_{i_{t'+1},\ldots,i_t,i_{m+1}} \vee \theta' \vee \bigvee_{t'<f\leq t} F_{i_f}$ logically entails $\bigvee_{f\leq t'} G_{i_f} \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f} \vee \theta' \vee \bigvee_{f\leq t} F_{i_f}$. Also, for every $f \leq t'$ we have $\neg G_{i_f} \vee F_{i_f}$ in $\mathcal{B}$. We get that

$$(\bigvee_{f\leq t'} G_{i_f} \vee \bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f} \vee \theta' \vee \bigvee_{f\leq t} F_{i_f}) \wedge \bigwedge_{f\leq t'} (\neg G_{i_f} \vee F_{i_f})$$

entails $\bigvee_{v'<f\leq v} l_{i_f} \vee \bigvee_{w<f\leq u} l_{i_f} \vee \theta' \vee \bigvee_{f\leq t} F_{i_f})$. If $\theta' = \mathit{FALSE}$, then this formula subsumes $\bigvee_{f\leq u} F_{i_f}$. On the other hand, if $\theta' = \bar{G}$, then, the same formula logically entails $\bigvee_{t<f\leq u} l_{i_f} \vee \bigvee_{f\leq t} F_{i_f})$ because $\neg(\bigvee_{f\leq t} G_{i_f}) \models \bigvee_{t<f\leq u} l_{i_f}$ and $\bar{G} \models \neg(\bigvee_{f\leq t} G_{i_f})$.

Thus, we are done because $\mathcal{B}$ includes $\widetilde{G}_{i_{t'+1},\ldots,i_t} \vee \theta' \vee \bigvee_{t'<f\leq t} F_{i_f}$, so $\mathcal{B} \models \bigvee_{f\leq u} F_{i_f}$ as needed. □

A.5 Proof of Corollary 4.5

Proof. We filter each of the literals separately and then combine the results. This is justified by Theorems 3.2,3.4, where we apply distribution for conjunction and disjunction recursively until we get to single literals.

To filter a literal l_i in φ we need to find the strongest formulae of the form $G_{i_1} \vee \ldots \vee G_{i_u}$ for $i_1, \ldots, i_u \in \{1, \ldots, l\} \cup \mathcal{C}(l_i)$ that are implied by l_i. This can be done by enumerating all disjunctions of this form. There are 2^{l+2} such combinations because there are a total of $l + 2$ rules for every literal (there is one completion rule relevant for every literal, and we add r_{m+1}). For every such a disjunction, checking that l_i implies it can be done in time 2^t by exhaustive enumeration of all truth assignments on the t symbols comprising the preconditions of a. Finally, computing $\mathcal{B}$ can be done in time $O(2^{l+t})$ as well. □

A.6 Proof of Corollary 4.6

Proof.

Lemma A.2. *For a formula φ, with $L(\varphi) \subseteq \mathit{Eff}(a)$ and $\varphi \models \neg\bar{G}$,*

$$Filter[a](\varphi) = \bigwedge_{\substack{i_1,\ldots,i_u \in \{1,\ldots l\},\\ \varphi \Rightarrow G_{i_1} \vee \ldots \vee G_{i_u}}} (F_{i_1} \vee \ldots \vee F_{i_u}) \bigwedge \mathcal{B}$$

We filter each of the literals separately and then combine the results using the recursive algorithm presented in Corollary 4.5. However, here we break only disjunctions and leave the conjunctions untouched.

Every disjunct D_i is a conjunction of literals $l_1 \wedge \ldots \wedge l_u$. For each such conjunction, asserting it and then testing whether $G_{i_1} \vee \ldots \vee G_{i_u}$ follows can be done in time 2^t. We can compute the DNF formula by distributing formula 5 over conjunctions.

We are left to prove that there are no more than $\binom{d+c}{(d+c)/2} \cdot s$ resulting disjuncts in this filtering. We do so by showing that the filtering of every D_i is a DNF with at most $\binom{d+c}{(d+c)/2}$ disjuncts. We divide the proof into three cases.

If $L(D_i) \cap \mathit{Eff}(a) = \emptyset$, then the only disjuncts coming as a result of filtering D_i are those coming from the conjunction of D_i and $\mathcal{B}$. The largest number of combinations of n elements that do not subsume each other (no combination is a subset of another) is $\binom{n}{n/2}$. Thus, the DNF form of $\mathcal{B}$ can have at most $\binom{d}{min(d/2,l)} \cdot c^l$ disjuncts because there are at most l choices from the F_i's in creating disjuncts from $\mathcal{B}$ and there are c disjuncts to choose from in each occurrence of a $\neg G_i$ (and we have a different G_i in each conjunct of $\mathcal{B}$). Thus, the total number of disjuncts in the filtering of D_i is at most $\binom{d}{min(d/2,l)} \cdot c^l$.

If $L(D_i) \subseteq \mathit{Eff}(a)$, then there are three cases according to the relationship between D_i and $\bar{G}$. If $D_i \models \bar{G}$, then the filtering is $D_i \wedge \mathcal{B}$, as above. Thus, in this case, the proof is done. If $D_i \models \neg\bar{G}$, then the filtering is in the form of equation (5) but with no completion rules involved. In this case, the second part of $\mathcal{B}$ is subsumed by one of the disjunctions in the first part of equation (5). Thus, we are left with a choice among the d disjunctions in the F_i's and then a choice of at most l elements among the disjuncts of the G_i's. This leaves us with at most $\binom{d}{d/2} \cdot c^l$ disjuncts in the filtering of D_i. Finally, if $D_i \not\models \bar{G}$ and $D_i \not\models \neg\bar{G}$, then we get that the filtering of D_i is the disjunction of the filtering of $D_i \wedge \bar{G}$, $D_i \wedge \neg\bar{G}$. This is $D_i \wedge \mathcal{B} \vee (5)$ for some formula of the form (5) that does not include completion rules. Using the previous cases we get that there are at most $2\binom{d}{d/2} \cdot c^l$ disjuncts in this case.

In the third case $L(D_i) \cap \mathit{Eff}(a), L(D_i) \setminus \mathit{Eff}(a) \neq \emptyset$. Here, the same argument as in the last paragraph shows that we have at most $2\binom{d}{d/2} \cdot c^l$ disjuncts in the filtering of D_i.

In the case of an exhaustive set of rules we always know that $D_i \models \neg\bar{G}$, so the factor 2 above never applies. □

A.7 Proof of Theorem 4.7

Proof. Take C to be one of the clauses of φ. We filter it separately from the rest and show that the filtering of C with a is a k-CNF formula. This will imply that the filtering of the conjunction of clauses is also a k-CNF formula, when we allow a decomposition over conjunction (as in Theorem 3.4).

We look at the clauses that result from equation (5). First we look at $\mathcal{B}$. Every parenthesized formula in $\mathcal{B}$ is of the form $(\neg G_i \vee F_i)$ or $(\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f\leq u} F_{i_f})$.

The first kind of formula can be written in k-CNF by noticing that if $F_i = f_1 \wedge \ldots \wedge f_e$, then $G_i \vee (f_1 \wedge \ldots \wedge f_e) \equiv (G_i \vee f_1) \wedge \ldots \wedge (G_i \vee f_e)$. Since $|L(G_i)| - min(1, |L(F_i) \setminus L(G_i)|) \leq k$, we get that each $(G_i \vee f_j)$ can be written in k-CNF. Thus $(\neg G_i \vee F_i)$ can be written in k-CNF.

For the second kind of formula, we check the different conditions. First, if $l = 1$, then we never get a formula of this kind (only of the first kind). If $|\bigcup_{i\leq l} L(G_i) \cup \mathit{Eff}(a)| \leq k$, then $|L(\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f\leq u} F_{i_f})| \leq k$, so $\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f\leq u} F_{i_f}$ can be written in k-CNF. Finally, if $|\bigcup_{i\leq l} L(G_i) \setminus \mathit{Eff}(a)| + l \leq k$, then $|L(\widetilde{G}_{i_1,\ldots,i_u})| \leq k-l$, and $\widetilde{G}_{i_1,\ldots,i_u}$ can be written in $(k-l)$-CNF. Since there are l rules and every F_i is a conjunction of literals, the disjunction $\bigvee_{f\leq u} F_{i_f}$ can be written in l-CNF. Thus, the combined formula, $\widetilde{G}_{i_1,\ldots,i_u} \vee \bigvee_{f\leq u} F_{i_f}$ can be written in $((k-l)+l) = k$-CNF. Thus, $\mathcal{B}$ can be written in k-CNF.

For the main part of the filtered clause in equation (5), we look at one formula that is part of the main conjunction, i.e., $(F_{i_1} \vee \ldots \vee F_{i_u})$. C is a clause with k literals, so $C = p_1 \vee \ldots \vee p_v \vee q_1 \vee \ldots \vee q_{k-v}$, for $\{p_j\}_{j\leq v}, \{q_j\}_{j\leq k-v}$ literals such that $L(\{p_j\}_{j\leq v}) \cap \mathit{Eff}(a) = \emptyset$ and $L(\{q_j\}_{j\leq k-v}) \subseteq \mathit{Eff}(a)$, for $0 \leq v \leq k$ (i.e., any one of the sets may be empty).

For all $j \leq v$, assume that $F_{i_j} = p_j$ (if p_j does not appear in $(F_{i_1} \vee \ldots \vee F_{i_u})$, then the argument is simpler, as will be apparent shortly (it implies that the resulting formula is even shorter than our worst-case scenario)).

Now, for every $j > v$, rule i_j refers to an original rule of a or the rule about $\bar{G}$ (but not a completion rule). Since $\forall j \leq v\ L(F_{i_j}) \cap \mathit{Eff}(a) = \emptyset$, we get that $(q_1 \vee \ldots \vee q_{k-v}) \models G_{i_{v+1}} \vee \ldots \vee G_{i_u}$. Thus, for every $j \leq k-v$, $q_j \models G_{i_{v+1}} \vee \ldots \vee G_{i_u}$. From our theorem's second assumption we get that $\models \bigvee_{j\leq u} G_{i_j}$ or for every $j \leq k-v$ there is $j' \leq u$ such that $q_j \models G_{i_{j'}}$.

If $\models \bigvee_{v<j\leq u} G_{i_j}$, then $\bigvee_{v<j\leq u} F_{i_j}$ is a consequence of the filtering that subsumes $\bigvee_{j\leq u} F_{i_j}$. Now we go back to the first assumption. If $l = 1$, then $\bigvee_{j\leq u} F_{i_j}$ is a conjunction of literals because F_1 is a conjunction of literals and there is only one effect rule. If $|\bigcup_{i\leq l} L(G_i) \cup \mathit{Eff}(a)| \leq k$, then $|\mathit{Eff}(a)| \leq k$, so $\bigvee_{v<j\leq u} F_{i_j}$ can be written in k-CNF. Finally, if $|\bigcup_{i\leq l} L(G_i) \setminus \mathit{Eff}(a)| + l \leq k$, then $l \leq k$, so $\bigvee_{j\leq u} F_{i_j}$ is a disjunction of $l \leq k$ disjuncts (conjunctions of literals), which can be represented in k-CNF (in fact, l-CNF).

For the last case, we examine what happens when for every $j \leq k-v$ there is $j' \leq u$ such that $q_j \models G_{i_{j'}}$. Thus, $C \models \bigvee_{j\leq v} p_j \vee \bigvee_{j\leq k-v} G_{i'_j}$, which implies that the filtering of C implies $\bigvee_{j\leq v} p_j \vee \bigvee_{j\leq k-v} F_{i'_j}$. This is a formula that subsumes $\bigvee_{j\leq u} F_{i_j}$, and it is also a disjunction of k terms, thus representable in k-CNF.

Thus, we showed that in all the cases either the formula is subsumed by a k-CNF formula or is itself logically equivalent to a k-CNF formula. This shows that the filtering of C can be written as a k-CNF formula. The filtering of φ follows using Theorems 3.2 and 3.4. □

A.8 Proof of Theorem 4.9

Proof. Every element of a disjunction $G_{i_1} \vee \ldots \vee G_{i_u}$ is either the precondition of an effect rule, a single literal (completion rule for literals not in $\mathit{Eff}(a)$) or a conjunction of a single literal with $\bar{G}$ (completion rule for literals in $\mathit{Eff}(a)$). Thus, each such disjunction is equivalent to a conjunction of (1) $\bar{G}$, (2) a disjunction of a set of literals and (3) a disjunction of preconditions of effect rules. The latter has at most $l \cdot d$ disjuncts each with at most t literals. Taking one literal from each of those disjunct leads to a representation of the latter part as a conjunction of $t^{l\cdot d}$ clauses. $\bar{G}$ has $l \cdot d$ conjuncts, each of at most t literals. Enumerating all the clauses that result from the combination of those parts yields a conjunction of at most $t^{l\cdot d} \cdot (l \cdot d + 1)$ clauses (we add 1 because if $\bar{G}$ participates, then $l \wedge \bar{G} \vee \ldots$ breaks into $(l \vee \ldots) \wedge (\bar{G} \vee \ldots)$).

Now, always if $\varphi \models A \wedge B$, then $\varphi \models A$ and $\varphi \models B$. Thus, if A and B are clauses, then $\varphi \models A \wedge B$ iff there are prime implicates of φ such that one subsumes A and the other subsumes B. More generally, if A_g, $g \leq z$, are clauses, then $\varphi \models \bigwedge_{g\leq z} A_g$ iff there are z prime implicates of φ, $C_{j_1}, \ldots, C_{j_z}$ such that $C_{i_g} \models A_g$, for all $g \leq z$.

We get that for z being the number of clauses that represent $G_{i_1} \vee \ldots \vee G_{i_u}$,

$$
\begin{aligned}
&Filter[a](\varphi) = \\
&\quad \textstyle\bigwedge_{i_1,\ldots,i_u\leq m,\ \varphi\models\bigvee_{f\leq u} G_{i_f}} (\bigvee_{f\leq u} F_{i_f}) \equiv \\
&\quad \textstyle\bigwedge_{i_1,\ldots,i_u\leq m,\ (\bigwedge_{g\leq z} C_{j_g})\models(\bigwedge_{g\leq z} A_g)} (\bigvee_{f\leq u} F_{i_f}) \equiv \\
&\quad \textstyle\bigwedge_{\exists j_1,\ldots,j_z; i_1,\ldots,i_u\ (\bigwedge_{g\leq z} C_{j_g})\models(\bigvee_{f\leq u} G_{i_f})} (\bigvee_{f\leq u} F_{i_f}) \equiv \\
&\quad \textstyle\bigwedge_{j_1,\ldots,j_z\leq s} \bigwedge_{i_1,\ldots,i_u\leq m\ (\bigwedge_{g\leq z} C_{j_g})\models(\bigvee_{f\leq u} G_{i_f})} (\bigvee_{f\leq u} F_{i_f}) \equiv \\
&\quad \textstyle\bigwedge_{j_1,\ldots,j_z\leq s} Filter[a](\bigwedge_{g\leq z} C_{j_g})
\end{aligned}
$$

This shows that every conclusion of $Filter[a](\varphi)$ is a conclusion from some z clauses C_i for $i \leq s$. Thus, the conjunction of filtering the conjunction of every z clauses from φ results in a formula equivalent to the filtering of φ.

Now, we know that $Filter[a](\varphi)$ includes every clause C_i that satisfies $L(C_i) \cap \mathit{Eff}(a) = \emptyset$ (follows from (3)). It follows that we can eliminate from consideration in $\bigwedge_{j_1,\ldots,j_z\leq s} C_{j_g}$ those clauses C_i that satisfy $L(C_i) \cap (\mathit{Eff}(a) \cup Pre(a)) = \emptyset$, for $Pre(a) = \bigcup_{i\leq l} L(G_i)$, because every clause that they may imply is subsumed by the clause $G_{i_1} \vee \ldots \mathcal{G}_{i_u} = C_i$, for $G_{i_1}, \ldots, G_{i_u}$ being the set of preconditions of completion rules for literals that appear in C_i. Thus, these C_i's can be filtered separately from the rest. Call this set of indexes $\mathcal{I}_1$ (i.e., $\forall i \in \mathcal{I}_1\ L(C_i) \cap (\mathit{Eff}(a) \cup Pre(a)) = \emptyset$).

On the other hand, every prime implicate clause that we choose for the conjunction $\bigwedge_{j_1,\ldots,j_z\leq s} C_{j_g}$ determines a set of literals outside $Pre(a)$ that it can subsume, and that should be in the disjunction $G_{i_1} \vee \ldots \vee G_{i_u}$. We get that every conjunction of prime implicate clauses determines (uniquely) a

disjunction $G_{i_1} \vee ... \vee G_{i_u}$ that it entails (it may entail a stronger formula, but the way we match clause sets with these disjunctions allows us the slack of knowing that if this formula is not the strongest we can entail, then there is another conjunction that will entail it for us). In particular, let $\mathcal{G}$ be the set of $g \leq z$ such that $L(C_{i_g}) \cap Pre(a) = \emptyset$. Then, $\bigwedge_{g \leq z} C_{j_g} \models G_{i_1} \vee ... \vee G_{i_u}$ implies that if $g \in \mathcal{G}$, then C_{i_g} subsumes $G_{i_1} \vee ... \vee G_{i_u}$. Thus, Filtering C_{i_g} implies something stronger than the one selected for $\bigwedge_{g \leq z} C_{j_g} \models G_{i_1} \vee ... \vee G_{i_u}$. Thus, we can filter all C_i's with $L(C_i) \cap Pre(a) = \emptyset$ separately from the rest.

We get that we can filter all clauses C_i such that $L(C_i) \cap (Pre(a) \cap \mathit{Eff}(a)) = \emptyset$ separately, and consider only clauses C_i with $L(C_i) \cap (Pre(a) \cap \mathit{Eff}(a)) \neq \emptyset$ in the second stage (when we choose z clauses and filter them together).

As above, for every disjunction $G_{i_1} \vee ... \vee G_{i_u}$ entailed by φ there is such a choice of z prime implicate clauses that entails this disjunction. The choice of those clauses is unique, as described above. Thus, when we choose a set of clauses it is enough to ignore the part of those clauses that is in $\overline{Pre(a)}$ and use the rest to find the effects that they generate in $Pre(a)$. In other words, it is enough to iterate over choices of disjunctions among $G_1, ..., G_l, \bar{G}$ and the literals in $Pre(a)$.

Here is the algorithm. For every choice from $G_1, ..., G_l, \bar{G}$ and the set of literals in $Pre(a) \setminus \mathit{Eff}(a)$ find the CNF representation of the disjunction of those G_i's (there are at most z clauses, as discussed above). For every clause C in this CNF select nondeterministically a prime implicate clause C_i whose restriction to $Pre(a)$ subsumes C. The joint selection implies the disjunction of those G_i's and some literals from $\overline{Pre(a)}$. Add the proper disjunction $\bigvee_{h \leq j}(F_{i_h} \vee (C_{i_h} \cap \overline{\mathit{Eff}(a)}))$ to the result of the filtering. Also, if $\bar{G}$ is in the set of original G_i's that we selected, then let A be the set of literals of $\mathit{Eff}(a)$ that appear in some C_i. Nondeterministically select a subset of A and let D be the disjunction of those literals. If the conjunction of our chosen clauses implies the disjunction of G_i, when we replace $\bar{G}$ with D, then add the disjunction of the original F_i and D to the filtering. (This takes care of the case that we prove the disjunction of some effect-rule preconditions and the preconditions of some $\mathit{Eff}(a)$-literals completion rules.)

This algorithm has $2^{l \cdot |Pre(a) \cup \mathit{Eff}(a)|}$ possible implications to check. Each implication involves generating the z-sized CNF, and every clause may match s subsuming clauses from φ. For each combination of such clauses from φ we generate their implied result in the filtered formula. Thus, the time to compute all the implied clauses from this filtering algorithm is $O(2^{l \cdot |Pre(a) \cup \mathit{Eff}(a)|} \cdot (s^z + z))$.

For the case of $\models \neg\bar{G}$ we know that there is no need to include $\bar{G}$ in the selection algorithm above. Also, we can omit the second part of the algorithm above. This means that in this case there are only 2^l possible implications to check. Consequently, the time of the resulting algorithm in this case is $O(2^l \cdot (s^z + z))$. □

A.9 Proof of Corollary 4.11

Proof. Notice that Corollary 4.10 and the conditions of our current theorem imply that equation (6) holds with $z = 1$, i.e., that we can filter each clause separately. Theorem 4.7 implies that the result of filtering each clause of φ is a k-CNF formula. Thus, the filtering of φ is also a k-CNF formula. □

A.10 Proof of Theorem 4.12

Proof. First, we notice that the following hold in a STRIPS domain for every action a:

$$Filter[a](\mathit{TRUE}) \equiv F_1 \wedge Cn^{\overline{\mathit{Eff}(a)}}(G_1).$$

Let $T_a = Filter[a](\mathit{TRUE})$. We get that for a literal l,

$$Filter[a](l) \equiv \begin{cases} T_a & l \in \mathcal{L}(\mathit{Eff}(a)) \\ T_a \wedge l & l \notin \mathcal{L}(\mathit{Eff}(a)) \end{cases}$$

Thus, for a set of literals $l_1, ..., l_k$ we get that equation (7) holds.

Thus, the main computation involved in filtering a k-CNF formula is computing T_a. In turn, the only computation needed there is finding the k-CNF form of $Cn^{\overline{\mathit{Eff}(a)}}(G_1)$. This can be done in time $O(2^t)$ and we need to do it only once per action (in fact, we can do that prior to the computation of filtering, and speed up the real-time filtering of the belief state).

Then, we use a simplified version of the algorithm described in Figure 1. We break the k-CNF formula into its clauses, and filter each of them separately using the equivalence above. This is done in time $O(k)$ per clause, thus yielding a total time of $O(s \cdot k)$.

To see that we maintain a k-CNF all is needed is to notice that each conjunction that results from filtering a clause in φ is a clause in F_1 (which has at most k literals) or a clause with literals in $L(\overline{\mathit{Eff}(a)}) \cap L(G_1)$ (which has at most $t \leq k$ literals). The only other clause that can be in the result of the filtering is a clause C_i with $L(C_i) \cap \mathit{Eff}(a) = \emptyset$. This C_i has k literals because φ is a k-CNF formula. □

A.11 Proof of Theorem 4.13

Proof. We need to show that $\bigwedge_{i \leq s} Filter[a](C_i) \models Filter[a](\bigwedge_{i \leq s} C_i)$, and the opposite direction will follow from Theorem 3.2. Equivalence (3) implies that we need to show only $\bigwedge_{i \leq s} Filter[a](C_i) \models \bigvee_{i \in I} F_i$ whenever $\bigwedge_{i \leq s} C_i \models \bigvee_{i \in I} G_i$ and I is as appropriate in Equivalence (3).

The proof of Theorem 4.9 shows that clauses C_i with $L(C_i) \cap (Pre(a) \cup \mathit{Eff}(a)) = \emptyset$ can be filtered separately. Call φ^2 the conjunction of clauses from φ that include propositional symbols from $Pre(a) \cup \mathit{Eff}(a)$, and call φ^1 the conjunction of the rest of the clauses. Also, every disjunction $\bigvee_{f \leq u} G_{i_f}$ in Equivalence (3) is implied by some conjunction of prime implicate clauses from φ^2.

Now, G_1 always holds because a is a (successful) STRIPS action, so we always know after the action that F_1 holds. The only disjunctions $\bigvee_{f \leq u} G_{i_f}$ that are not subsumed by G_1 are those that do not include it. The G_i's that are not G_1 and that might appear in such disjunctions are only preconditions of completion rules in $\mathcal{C}(L(\varphi))$. In addition, $\bar{G} = \neg G_1$, and

G_1 is consistent with φ (**we assume that our belief state is nonempty and that our transition model is correct**). Thus, whenever we can prove from φ^2 a disjunction $\bigvee_{f \leq u} G_{i_f}$ in which $\bar{G}$ participates, we can always prove a stronger formula in which $\bar{G}$ does not participate (because whatever we prove has to be consistent with G_1, i.e., with $\neg\bar{G}$).

Thus, we know that we can compute the result of the filtering of φ^2 by looking at disjunctions $\bigvee_{f \leq u} G_{i_f}$ that involve only completion rules for literals in $L(\varphi^2) \setminus \mathit{Eff}(a)$. Thus, each G_{i_f} is a literal that does not appear in $\mathit{Eff}(a)$, and $\bigvee_{f \leq u} G_{i_f}$ is a single clause C.

If $\varphi^2 \models C$, then there is a clause C_i in φ such that C_i subsumes C because φ is in PI-CNF. Thus, for some C_i in φ, $Filter[a](C_i)$ will imply the matching clause sanctioned by (3). We get that every result of $Filter[a](\bigwedge_{i \leq s} C_i)$ is implied by $\bigwedge_{i \leq s} Filter[a](C_i)$, i.e., $\bigwedge_{i \leq s} Filter[a](C_i) \models Filter[a](\bigwedge_{i \leq s} C_i)$. □

A.12 Proof of Corollary 4.14

Proof. Theorem 4.13 guarantees that we can filter each of the clauses separately and conjoin the result. From equivalence (7) we know that the filtering of each clause is equivalent to $Filter[a](\mathit{TRUE})$ conjoined with either that clause or *TRUE*.

First, look at the clauses that are kept from φ and those prime implicates that are given by $Cn^{\overline{\mathit{Eff}(a)}}(G_1)$. For every original clause that is kept for the new belief state (after filtering with a), either it is a prime implicate in the new belief state or it is subsumed by a prime implicate of $Cn^{\overline{\mathit{Eff}(a)}}(G_1)$. If it is not subsumed, then it is a prime implicate by definition. If it is subsumed, then the clause that subsumed it is a prime implicate by definition. A similar situation holds in the other direction (clauses of $Cn^{\overline{\mathit{Eff}(a)}}(G_1)$). In either case, those clauses that are left are in k-CNF. Since they are all in $\mathcal{L}(\overline{\mathit{Eff}(a)})$, there is nothing else that can subsume them (the only other thing we add is F_1).

Finally, we look at the prime implicates of F_1. They are in the language $\mathcal{L}(\mathit{Eff}(a))$. For every original clause that intersects with this language, that clause is not transferred to the new belief state. Thus, there is no clause that can subsume any prime implicate of F_1. Since every prime implicate of F_1 has at most k literals, the resulting belief state is a prime implicate belief state that is in k-CNF. □

A.13 Proof of Theorem 3.7

Proof. Assume that for every filtering problem, T, there is a belief state representation such that $|\sigma_i| \leq poly(|T| * i)$, as in the statement of the theorem. Let A, B be two propositional formulae (in CNF) such that $A \models B$. ≪Make sure CNF is OK in Sipser's paper. –Eyal≫

We now create a dynamic system and a filtering problem T such that $|\sigma_i| \leq poly(|T| * i)$ and provide an interpolant for A, B that is of size $\leq poly(|A| + |B|)$.

First, we map the propositions of A and B to time $0, 1$, respectively. Let $L(B)_0$ be a set of new symbols that correspond to the symbols in $L(B)$. Let the propositions defining the state at time 0 be the symbols that appear in $L(B)_0 \cup L(A) \setminus L(B)$. Let $L(B) \cup L(A)$ represent the state at time 1.

Now, we map some of the formulae of A, B to the time steps. Let $\sigma_0 = A \setminus \mathcal{L}(B)$ be the belief state description at time 0 and let $Q1 = (A \cap \mathcal{L}(B)) \Rightarrow B$ be a query about the state at time 1.

We map the remaining formulae from A to effect propositions. Let $E = A \setminus (\sigma_0 \cup Q1)$ be the effect axioms used in performing an action a_1: Every clause of A can be seen as an effect axiom by viewing those parts of the clause that are in $L(A) \setminus L(B)$ to be the preconditions and the parts of the clause that are in $L(A) \cap L(B)$ as the effects. We create a single action, a, and give all the effect axioms to a. If $F \vee G \in E$, for F a clause in $\mathcal{L}(A)$ and G a clause in $\mathcal{L}(B)$, then add "a **causes** F **if** $\neg G$" to our filtering problem.

For T there is a belief state representation such that $|\sigma_1| \leq poly(|T|)$. But $|T| = O(|A \cup B|)$. Thus, $|\sigma_1| \leq poly(|A| + |B|)$. Let σ_1' be the syntactic translation of σ_1 into the vocabulary $L(A) \cup L(B)$. We know that σ_1' must logically imply every interpolant of $A \models B$ because σ_1' implies the set of implicates of A in $L(A) \cap L(B)$ (σ_1 is all that we know about the belief state at time 1 that logically follows from A's translation into our dynamic system setup (this translation is the state at time 0 and the effect axioms).

Thus, $A \models \sigma_1'$ and $\sigma_1' \models B$ and $|\sigma_1'| \leq poly(|A| + |B|)$. Since our choice of A, B was arbitrary, this implies that NP ∩ co-NP ⊆ P/poly by Theorem 3.6. □

A.14 Proof of Corollary 3.8

Proof. First, notice that the proof of Theorem 3.7 built a system of size equal to the total size of A, B from Theorem 3.6 and the application of a single action was enough to make the state exponentially large.

Take this system and separate each of the action's effects (at most n fluents) into separate actions' effects. First, for every effect rule r_i of action a, r_i corresponds to a clause. Divide the clause into effect clause eff_i and precondition clause pre_i, each of length l, say. Thus, pre_i is of the form $\mathsf{pre}_i^1 \wedge ... \wedge \mathsf{pre}_i^l$, and eff_i is of the form $\mathsf{eff}_i^1 \vee ... \vee \mathsf{eff}_i^l$ Add a new set of fluents, $f_1^i, ..., f_l^i$, $g_1^i, ..., g_l^i$, and new actions $a_1^i, ..., a_l^i$, $b_1^i, ..., b_l^i$. For a_j^i we have the precondition $f_{j-1}^i \wedge \mathsf{pre}_i^j$ and the effect f_j^i. We set f_0^i to be TRUE in the initial belief state. For b_j^i, we have the preconditions f_l^i and the effect $f_j^i \vee \mathsf{eff}_i^j$. We set f_l^i to FALSE in the last belief state.

Do so for all effect rules of action a. Now, join all the rules a_j^i for all i (holding j) into a new action a_j (this has to be done carefully, to ensure that we always add all that we know about a single fluent at the same time, so that we do not override previously asserted effects (as our semantics dictates)). Applying $a_1, ..., a_l$ results in the same belief state as after performing a in the original system, and the new fluents are fully known (are set to TRUE or FALSE in the belief state). Thus, the representation of this belief state is exactly that of the belief state of the system from Theorem 3.7, so it is exponential in the sizes of φ_0 and D. □

References

[Boppana and Sipser, 1990] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In Jan van

Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 1: Algorithms and Complexity, pages 757–804. Elsevier/MIT Press, 1990.

[Cimatti and Roveri, 2000] Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[Craig, 1957] William Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *Journal of Symbolic Logic*, 22:250–268, 1957.

[Doherty *et al.*, 1998] Patrick Doherty, Witold Lukaszewicz, and Ewa Madalinska-Bugaj. The PMA and relativizing change for action update. In *Principles of Knowledge Representation and Reasoning: Proc. Sixth Int'l Conference (KR '98)*, pages 258–269. Morgan Kaufmann, 1998.

[Ferraris and Giunchiglia, 2000] Paolo Ferraris and Enrico Giunchiglia. Planning as satisfiability in nondeterministic domains. In *Proc. National Conference on Artificial Intelligence (AAAI '00)*, pages 748–753. AAAI Press, 2000.

[Fikes *et al.*, 1972] Richard Fikes, Peter Hart, and Nils Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3:251–288, 1972.

[Kalman, 1960] Emil Kalman, Rudolph. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[Kautz *et al.*, 1996] Henry Kautz, David McAllester, and Bart Selman. Encoding plans in propositional logic. In J. Doyle, editor, *Proceedings of KR'96*, pages 374–384, Cambridge, Massachusetts, November 1996. KR, Morgan Kaufmann.

[Liberatore, 1997] Paolo Liberatore. The complexity of the language A. *Electronic Transactions on Artificial Intelligence (http://www.etaij.org)*, 1(1-3):13–38, 1997. http://www.ep.liu.se/ej/etai/1997/002/.

[Lin and Reiter, 1997] Fangzhen Lin and Ray Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1-2):131–167, 1997.

[McIlraith, 1998] Sheila McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In Anthony G. Cohn, Lenhart Schubert, and Stuart C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proc. Sixth Int'l Conference (KR '98)*, pages 167–177. Morgan Kaufmann, San Francisco, California, 1998.

[Reiter, 2001] Raymod Reiter. *Knowledge In Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, 2001.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Simon and del Val, 2001] Laurent Simon and Alvaro del Val. Efficient consequence-finding. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI '01)*, pages 359–365. Morgan Kaufmann, 2001.

[Son and Baral, 2001] Tran Cao Son and Chitta Baral. Formalizing sensing actions: A transition function based approach. *Artificial Intelligence*, 125(1–2):19–91, 2001.

[Williams and Nayak, 1996] Brian C. Williams and P. Pandurang Nayak. A model-based approach to reactive self-configuring systems. In *Proc. National Conference on Artificial Intelligence (AAAI '96)*, pages 971–978. AAAI Press, 1996.

[Winslett, 1990] Mary-Anne Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.